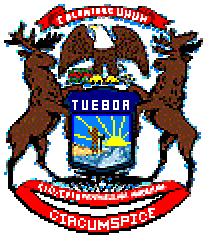


SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX A

GLOSSARY

Appendix A: Glossary

A

Acceptance criteria
The criteria that a software component, product, or system must satisfy in order to be accepted by the system owner or other authorized acceptance authority.

Acceptance process
The process used to verify that a new or modified software product is fully operational and meets the system owner's requirements. Successful completion of the acceptance process results in the formal transfer of the software product responsibilities from development to maintenance personnel.

Acceptance testing
Formal testing conducted to determine whether or not a software product or system satisfies its acceptance criteria and to enable the system owner to determine whether or not to accept the product or system.

Activity
A major unit of work to be completed in achieving the objectives of a software project. An activity incorporates a set of tasks to be completed, consumes resources, and results in deliverables. An activity may contain other activities in a hierarchical manner. All project activities should be described in the Project Plan.

Algorithm
A finite set of well-defined rules for the solution to a problem in a finite number of steps. Any sequence of operations for performing a specific task.

Allocated requirements
The subset of the system requirements that are to be implemented in the software components of the system.

Anomaly
Anything observed in the operation or documentation of software that deviates from expectations based on previously verified software products or documents.

Application
Software products designed to fulfill specific needs.

Assumption
A condition that is taken to be true without proof or demonstration.

Audit
An independent examination of a deliverable to assess compliance with specifications, standards, quality or security requirements, contractual agreements, or other predetermined criteria.

B

Baseline
A set of configuration items (software components and documents) that has been formally reviewed and agreed upon, that serves as the basis for further development, and that can be changed only through formal change control procedures.

Baselined requirements
The set of project requirements that have been approved and signed off by the system owner during the Requirements Definition Phase. The software product design is based on these requirements. The baselined requirements are placed under configuration control.

C

Code
Computer instructions and data definitions expressed in a programming language or in a form that is output by an assembler, compiler, or other translator.

Code generator
A software tool that accepts as input the requirements or design for a computer program and produces source code that implements the requirements or design.

Code review
A meeting at which software code is presented to project personnel, managers, users, or other functional areas for review, comment, or approval.

Component
One of the parts that make up a system. A component may be hardware, software, or firmware and may be subdivided into other components.

Computer-Aided Software Engineering (CASE)
The use of computers to aid in the software engineering process. May include the application of software tools for software design, requirements tracing, code production, testing, document generation, and other software engineering activities.

Configuration control
An element of configuration management consisting of the evaluation, coordination, approval/disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification.

Configuration control board
A group of people responsible for evaluating and approving/disapproving proposed changes to configuration items, and for ensuring implementation of approved changes.

Configuration item
An aggregate of hardware or software components that are designated for configuration management and treated as a single entity in the configuration management process.

Configuration management
See Software configuration management

Constraint
A restriction, limit, or regulation that limits a given course of action or inaction.

Cost estimate
A formal estimate of the cost to develop and support a project. Estimates should reflect all activities such as design, development, coding, distribution, service, and support of the product; staffing; training and travel expenses; subcontractor activities; contingencies; and cost for external services (e.g., technical documentation production and Quality Assurance audits and reviews).

D

Deliverable
Any tangible item that results from a project function, activity, or task. Examples of deliverables include process descriptions, plans, procedures, computer programs, and associated documentation, which may or may not be intended for delivery to the system owner and other project stakeholders. A deliverable that is identified in the Project Plan and is formally delivered to the system owner and other project stakeholders for review and approval.

Dependency
A relationship of one task to another where the start or end date of the second task is related to the start or end date of the first task.

Design
The process of defining the architecture, components, interfaces, and other characteristics of a software product or component.

Design specification
A document that describes the design of a software component, product, or system. Typical contents include architecture, control logic, data structures, input/output formats, interface descriptions, and algorithms.

F

Feasibility
The degree to which the requirements, design, or plans for a software product or system can be implemented under existing constraints.

Functional area
Any formally organized group involved in the development and maintenance of software or the support of development and maintenance efforts, or other group whose input is required to successfully implement a software project. Examples of functional areas include software engineering services, technical writing, quality assurance, security, and telecommunications.

Functional design phase
The period of time in the software lifecycle during which the designs for architecture, software components, interfaces, and data are created, documented, and verified to satisfy project requirements.

Functional requirement
A requirement that specifies a function that a software component, product, or system must be able to perform.

Functional test plan
A plan for testing each function across one or more units. The plan describes how the functional testing occurs and the test procedure/test cases that will be used. The plan includes procedures for creating the test environment that allows all functions to be executed; the entry and exit criteria for starting and ending the function-testing period; and the schedule followed for starting and ending each test.

Functional test procedures
Procedures for each function or combination of functions to be tested. Procedures fully describe how the function is tested. Expected output from each test procedure is identified to compare the planned output to actual output.

Functional testing
Testing conducted to evaluate the compliance of a software product with specified functional requirements. Testing that focuses on the outputs generated in response to selected inputs and execution conditions.

H

Hardware
Physical computer and other equipment used to process, store, or transmit computer programs or data.

Hierarchy
A structure in which components are ranked into levels of subordination.

I

Implementation requirements
A requirement that supports the development and maintenance concepts and approaches in the areas of operating environment, conversion, installation, training, and documentation.

Incremental development
A software development technique in which requirements definition, design, implementation, and testing occur in an overlapping, iterative (rather than sequential) manner, resulting in incremental completion of the overall software product.

Information engineering
A development methodology where models are created to improve the users' ability to understand and define the functions and flow of information within their organization. A business model is developed to identify the key areas of interest for the business, the tasks required for each area, and the activities that make up each task. The business model prioritizes and identifies top management goals and then establishes the information needs necessary to reach those goals. A data model is developed to describe the data and the relationships among data. The data model further divides the business model into user-defined relationships (e.g., entity relationship model).

Inspection
A static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems. Code inspection and design inspection are two types.

Integration testing
An orderly progression of testing in which software components are combined and tested to evaluate the interaction between them.

Integrity
The degree to which a software component, product, or system prevents unauthorized access to, or modification of, computer programs or data.

Interactive analysis and design
A development methodology that uses facilitated team techniques, such as Joint Application Development or Rapid Application Development, to rapidly develop project requirements that reflect the users' needs in terminology that the users understand. Group facilitation techniques are especially important when several user organizations have unique project requirements that are specific to their mission and goals.

Interface requirement
A requirement that specifies an external item with which a software product or system must interact, or that sets forth constraints on formats, timing, or other factors caused by such an interaction.

Interface testing
Testing conducted to evaluate whether software components pass data and control correctly to one another.

Interview technique
A technique for the identification, analysis, and documentation of the project requirements. The project team conducts a series of interviews with users to identify the users' perceived automated functional needs, analyzes the information gathered during the interviews, and develops the requirements.

K

Key process area
Software engineering processes identified by the Software Engineering Institute Capability Maturity Model where a project team should focus its efforts to achieve consistently high quality software products.

L

Lifecycle See Software lifecycle.

M

Maintenance
The process of supporting a software product or system after delivery to maintain operational status, correct faults, improve performance or other attributes, or adapt to a changed environment.

Menu-driven
Pertaining to a system or mode of operation in which the users direct the software through menu selections.

Methodology
A collection of methods, procedures, and standards that defines an integrated synthesis of engineering approaches to the development of a deliverable.

Milestone
A scheduled event for which an individual or team is accountable and that is used to measure progress.

Module
A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading. A logically separable part of a program.

Module testing
Testing of individual software modules or groups of related modules to verify the implementation of the design.

O

Organization
An organization is a unit with a company or other entity within which projects are managed as a whole. An operational definition for “organization” is the scope of an appraisal or process improvement effort. Organization analysis is necessary to define exactly what the scope will be. All projects within an organization share a common top-level manager and common policies.

P

Performance requirement
A requirement that imposes conditions on a functional requirement (e.g., a requirement that specifies the speed, accuracy, or memory usage with which a given function must be performed).

Phase
A partition of the software lifecycle that reduces a project to manageable size and represents a meaningful and measurable set of related tasks that are performed to obtain specific deliverables.

Planning phase
The initial phase in the software lifecycle during which the system owner/users' needs and expectations are identified, the feasibility of the project is determined, and the Project Plan is developed.

Platform
A specific computer and operating system on which a software product is developed or operated.

Portability
The ease with which a software component, product, or system can be transferred from one hardware or software environment to another.

Procedure
A written description of a course of action to be taken to perform a given task.

Process
An ordered set of steps performed for a given purpose. Processes define or control the development of the project deliverables. The use of processes will ensure a consistent methodology across all platforms in producing the lifecycle deliverables.

Programmers reference manual
A deliverable that provides information necessary to maintain or modify software for a given computer system. Typically described are the equipment configuration, operational characteristics, programming features, input/output features, and compilation or assembly features of the computer system.

Programming phase
The period of time in the software lifecycle during which a software product is created from the design specifications and testing is performed on the individual software units.

Project
An undertaking requiring concerted effort that is focused on developing or maintaining a specific software product or system. A project has its own funding, cost accounting, and delivery schedule.

Project lifecycle
The software lifecycle selected for the project and approved by the system owner and other project stakeholder(s).

Project manager
The individual with total business responsibility for all software activities of a project. The project manager directs controls, administers, and regulates a project.

Project notebook
A central repository of material pertinent to a project. Contents typically include all deliverables, memos, plans, technical reports, and related items.

Project plan
A document that describes the technical and management approach to be followed for a project. The plan typically describes the work to be done, the resources required, the methods to be used, the procedures to be followed, the schedules to be met, and the way the project will be organized. The plan includes a list of deliverables, actions required, and other key events needed to accomplish the project.

Project team
The project manager, analysts, programmers, and other staff assigned as the core group for a project. The project team may include representatives of the other functional areas (e.g., technical writer and telecommunications expert) responsible for contributing to the development, installation, and maintenance of the software product.

Project Test Plan
Defines all test activities required to assure that the software product will perform satisfactorily for all users. As a minimum, the plan should include descriptions for unit testing, integration testing, system testing, and acceptance testing.

Prototyping
A technique for developing and testing a preliminary version of the software product (either as a whole or in modular units) in order to emulate functionality without such encumbering features as error handling, help messages, security controls, and other utilities that are not part of the design logic. This allows the project team to test the overall logic and workability of required functions and provides a model by which the project team and users can jointly determine if the software requirements meet the intended objectives. Prototyping is often used in conjunction with interactive analysis and design techniques.

Pseudocode
A combination of programming language constructs and natural language used to express a computer program design.

R

Rapid prototyping
A type of prototyping in which emphasis is placed on developing prototypes earlier in the development process to permit early feedback and analysis in support of the development process.

Reference
A document(s) or other material that is useful in understanding more about an activity.

Regression testing
Selective retesting of a software component to verify that modifications have not caused unintended effects and that the software component still complies with its specified requirements.

Reliability
The ability of a software component to perform its required functions under stated conditions for a specified period of time.

Requirement
A condition or capability needed by a system owner/user to solve a problem or achieve an objective. A condition or capability that must be met or possessed by the software product to satisfy a contract, standard, specification, or other formally imposed documents.

Requirements analysis
The process of studying system owner/user(s) needs to arrive at a definition of system, hardware, or software requirements.

Requirements definition phase
The period of time in the software lifecycle during which the requirements for a software product are defined and documented.

Requirements management
A Software Engineering Institute Capability Maturity Model key process area designed to establish a common understanding between the system owner/user and the project team regarding the system owner/users' software requirements. This understanding forms the basis for estimating, planning, performing, and tracking the project's activities throughout the lifecycle.

Requirements specification
A deliverable that specifies the manual and automated requirements for a software product in non-technical language that the system owner/users can understand. Typically included are functional requirements, performance requirements, and interface requirements. Describes in detail what will be delivered in the product release.

Retirement
Permanent removal of a system or software product from its operational environment.

Reusability
The degree to which a software module or other deliverable can be used in more than one computer program or software system.

Reverse engineering
A development methodology in which the software developments process is performed in reverse. The technique involves the examination of an existing software product that has characteristics that are similar to the desired product. Using the existing code as a guide, the requirements for the product are defined, analyzed, and abstracted all the way back to specifications. Any required code changes can be made based on a specification-like format. Ideally, the specifications would be edited and passed to a code generator that would trigger automatic documentation and revisions. Once testing is complete, the revised code is placed into production.

Risk
The possibility of suffering loss.

Risk management
An approach to problem analysis that is used to identify, analyze, prioritize, and control risks.

S

Software
Computer programs, procedures, and associated documentation and data pertaining to the operation of a software product or system.

Software configuration item
An aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.

Software configuration management
(1) A discipline that effectively controls and manages all modifications to a software component, product, or system. Technical and administrative processes and tools are used to identify and document the functional and physical characteristics of the configuration items, manage and track changes to those items, record and report change processing and implementation status, and verify compliance with specified requirements. (2) A Software Engineering Institute Capability Maturity Model key process area designed to establish and maintain the integrity of the software deliverables throughout the project's lifecycle.

Software lifecycle
The period of time that begins when a software product is conceived and ends when the software is retired. A network of phases and processes that function together to guide the development and maintenance of software products. Each process produces a set of deliverables as it moves through the lifecycle.

Software procurement
The process by which software is acquired. The process is defined by the procurement plan, which identifies business and/or technical requirements development, purchase methodology, budget, resources for a Joint Evaluation Committee (executive and core team), high-level testing plan, required approvals, and contracting methods. The software may be an off-the-shelf package that is a project product, or it may be testing, QA, operating system, database management, ...software that supports the project process.

Software project planning
A Software Engineering Institute Capability Maturity Model key process area designed to establish reasonable plans for performing software engineering and for managing the software project.

Software project management plan
The controlling document for managing a software project. The plan defines the technical and managerial functions, activities, and task necessary to satisfy the requirements of a software project, as defined in the project agreement. Synonymous with software development plan.

Software project tracking and oversight
A Software Engineering Institute Capability Maturity Model key process area designed to provide adequate visibility into actual project progress so that management can take effective actions when the project's performance deviates significantly from the plans.

Software quality assurance
A Software Engineering Institute Capability Maturity Model key process area designed to provide management with appropriate visibility into the software engineering processes being used by the project team and the deliverables being built.

Software system
A software product and the documentation, hardware, and communications needed to implement and operate the product and accomplish a specific function or set of functions.

Specification
A document that specifies in a complete, precise, verifiable manner the requirements, design, behavior, or other characteristics of a software component, product, or system.

Stakeholder
The Agency individual(s) with decision-making authority over a project or group of projects.

Standard
Mandatory requirements employed and enforced to prescribe a disciplined, uniform approach to software development and maintenance.

Structured analysis
An analysis technique that uses a graphical language to build models of software products or systems. The four basic features in structured analysis are data flow diagrams, data dictionaries, procedure logic representations, and data store structuring techniques.

System
A collection of hardware, software, firmware, and documentation components organized to accomplish a specific function or set of functions.

System design document
A deliverable that describes the solution to the automation task as described by the requirements. Contains sufficient detail to provide necessary direction for writing the Program Specifications and allows developers maximum technical freedom.

System design phase
A phase in the lifecycle model during which the designs for the software product architecture, software components, interfaces, and data are refined and expanded to the extent that the design is sufficiently complete to be implemented.

System owner
The organizational unit that funds and has approval authority for the project. Typically, system owners are also system users.

Systems development lifecycle
The State of Michigan's methodology that identifies the processes, activities, tasks, management responsibilities, and deliverables that are required for each software development and maintenance project. Deviations from the methodology are managed on a project by project basis. A key objective of the methodology is to provide measurable, repeatable processes to assure that project development and maintenance methodologies are consistent throughout the agency information technology environment.

System testing
Testing conducted on a complete, integrated software product or system to evaluate compliance with its specified requirements.

T

Task
The smallest unit of work subject to management accountability. A task is a well-defined work assignment for one or more project team members. Related tasks are usually grouped to form activities. A task is the lowest level of work division typically included in the Project Plan and Work Breakdown Structure.

Telecommunications
The science and technology of communications by electronic transmission of impulses, as by telephone or e-mail.

Test bed
An environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test.

Test case
A set of test inputs, execution conditions, and expected results that are developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

Test criteria
The criteria that a software component or product must meet in order to pass a given test.

Test design
Documentation specifying the details of the test approach for a software feature or combination of software features and identifying the associated tests.

Test documentation
Documentation describing plans for, or results of, the testing of a software component or product. Documents typically include test case specifications, test incident reports, test logs, test plans, test procedures, and test reports.

Test item
A software item that is the object of testing.

Test log
A chronological record of all relevant details about the execution and results of a test.

Test phase
The period of time in the software lifecycle in which the components of a software product are evaluated and integrated, and the software product is evaluated to determine whether or not the requirements have been satisfied.

Test plan
A document specifying the scope, approach, resources, and schedule of intended testing activities. The plan identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.

Test procedure
Detailed instructions for the setup, execution, and evaluation of the results for a given test case.

Test report
A document that describes the conduct and results of the testing carried out for a software component or product.

Testing
An activity in which a software component or product is executed under specified conditions, the results are observed and recorded, and an evaluation is made.

Traceability
The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor relationship to one another.

Transaction analysis
A technique used to derive structured charts for a software product that will process transactions. Transaction analysis is used to divide complex data flow diagrams into smaller, simpler data flow diagrams--one for each transaction that the product or system will process. Structure charts are developed from the simple data flow diagrams. The individual structure charts for the separate transactions are then combined to form one large structure chart that is very flexible and can accommodate user changes.

Transition plan
The system is transitioned into operational status. The transition plan contains materials, operating documents, and other pertinent records. These materials are turned over to the maintenance staff.

U

Unit
A separately testable element specified in the design of a computer software component. A software component that is not subdivided into other components.

Unit testing
Testing of individual hardware or software units or groups of related units. The isolated testing of each flowpath of code with each unit. The expected output from the execution of the flowpath should be identified to allow comparisons of the planned output against the actual output.

Usability
The ease with which a user can learn to operate, prepares inputs for, and interprets outputs of a software product.

User
The general population of individuals who use a software product or system. User activities can include data entry; read only; add, change and delete capabilities; querying; and report generation.

User interface
An interface that enables information to be passed between a user and hardware or software components of a computer system.

User manual
A document that presents the information necessary to use a software product to obtain desired results. Typically described are products or component capabilities, limitations, options, permitted inputs, expected outputs, possible error messages, and special instructions.

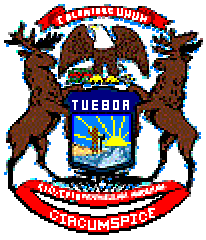
V

Validation
The process of evaluating software at the end of the software development process to assure compliance with established software and system requirements.

Verification
The process of evaluating a software product to determine whether or not the deliverables of a phase of the software lifecycle fulfill the requirements established during the previous phase.

W

Walkthrough
An analysis technique in which a team of subject matter experts review a segment of code or documentation, ask questions, and make comments about possible errors, violation of development standards, and other problems.



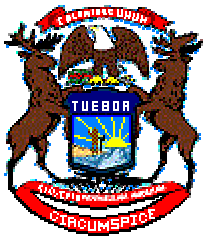
SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX B

LIST OF ABBREVIATIONS

Appendix B: List of Abbreviations

Abbreviation Code	Definition
ABC	Analysis of Benefits and Costs
ANSI	American National Standards Institute
CASE	Computer-Aided Software Engineering
FCA	Functional Configuration Audit
IBM	International Business Machines
IEEE	The Institute of Electrical and Electronics Engineers, Inc.
IP	Internet protocol
IPA	In-Phase Assessment
JEC	Joint Evaluation Committee
LAN	Local area network
PCA	Physical Configuration Audit
POC	Point of contact
SEI	Software Engineering Institute (at Carnegie-Mellon)
SOM	State of Michigan
SQA	Software Quality Assurance
Std	Standard
TCP	Transmission control protocol
VRD	Version Release Document
WAN	Wide area network



SYSTEMS DEVELOPMENT LIFECYCLE

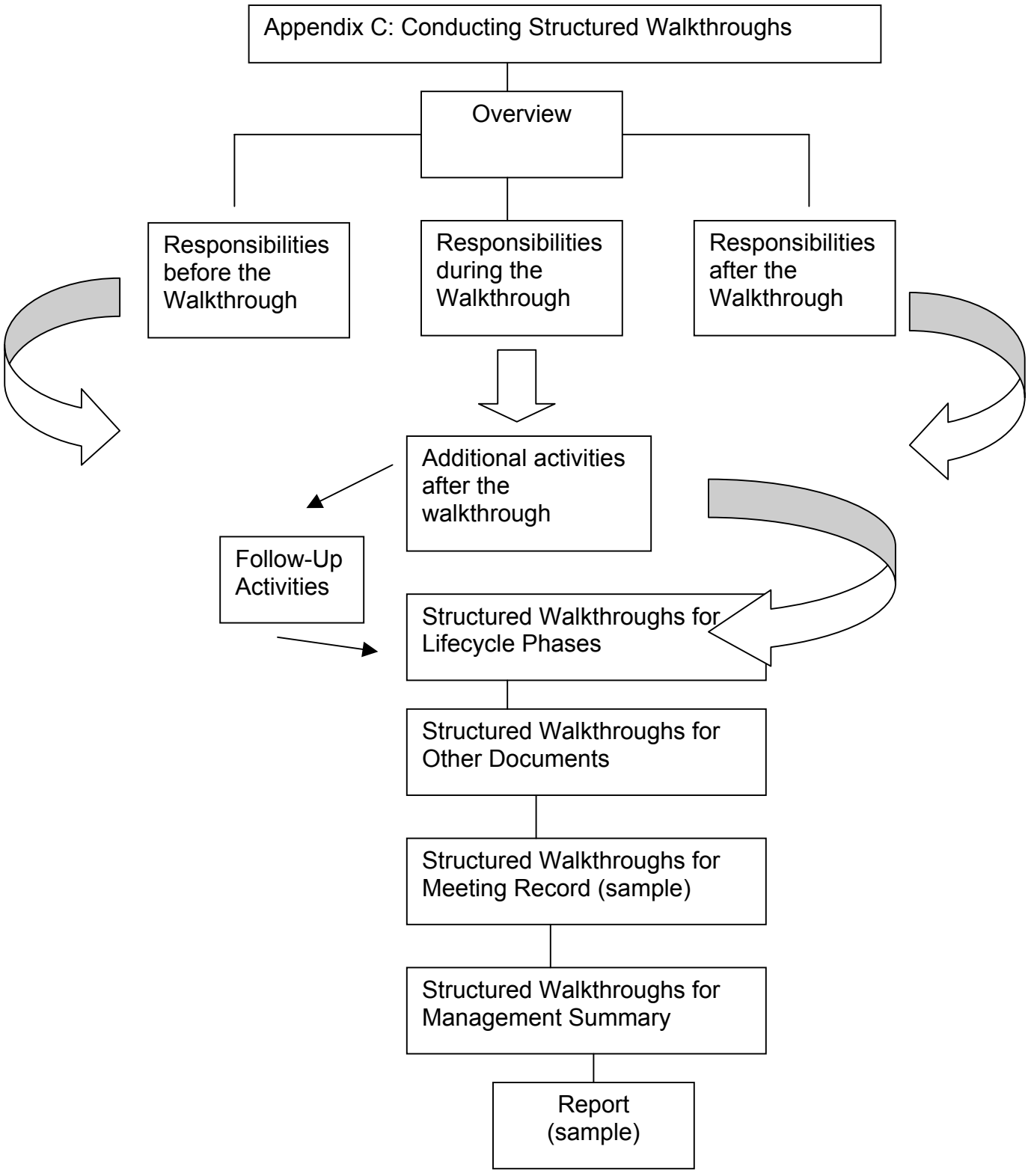
APPENDIX C

CONDUCTING STRUCTURED WALKTHROUGHS

Section C: Conducting Structured Walkthroughs

Table of Contents

Highlights of Phase



Responsibilities Before the Walkthrough

Purpose:

This process guide describes how to conduct a structured walkthrough during the lifecycle phases of software development projects, regardless of hardware platform.

Organization:

This process guide consists of the following sections:

- Overview
- Responsibilities Before the Walkthrough
- Responsibilities During the Walkthrough
- Responsibilities After the Walkthrough
- Additional Activities After the Walkthrough
- Follow-up Activities
- Structured Walkthroughs for Lifecycle Phases
- Structured Walkthroughs for Other Documents
- Structured Walkthrough Meeting Record (sample)
- Structured Walkthrough Management Summary Report (sample)

Description:

A structured walkthrough is an organized procedure for a group of peers to review and discuss the technical aspects of software development and maintenance deliverables. The major objectives in a structured walkthrough are to find errors and to improve the quality of the product. Errors typically occur as omissions or contradictions, flaws in logic, or inconsistencies in the deliverable style (e.g., poorly stated requirements and inefficient code).

Structured walkthroughs should not be used to discuss solutions for the errors that are found. *The basic purpose of a walkthrough is error detection, not error correction.* When the walkthrough is finished, the author of the work product is responsible for taking the necessary actions to correct the errors. The author may hold private conversations with reviewers or conduct follow-up meetings to discuss potential solutions.

Structured walkthroughs should be conducted during all phases of the software lifecycle. Walkthroughs can be conducted in various formats, with various levels of formality, and with different types of participants. In some cases, it might be useful and expedient to include software customers in walkthroughs. Management representatives do not participate in structured walkthroughs. Regardless of the variations in format and participants, the basic activity (peer review) and the major objectives (find errors and improve quality) of the structured walkthroughs remain the same.

Structured walkthroughs are appropriate for reviewing the technical accuracy and completeness of software development and maintenance deliverables, project management tools, and other types of documents (e.g., technical operating procedures). The walkthroughs should be scheduled to review small, meaningful pieces of work. The progress made in each lifecycle phase should determine the frequency of the walkthroughs.

General Information:

Benefits:

Structured walkthroughs provide the following benefits:

- Save time and money by finding and correcting errors earlier in the lifecycle.
- Provide value-added input from reviewers with different technical backgrounds, experience, and expertise.

Responsibilities Before the Walkthrough

General Information Continued:

- Validate and improve the related lifecycle deliverables. Keep the project team informed of the development or maintenance progress.
- Provide professional growth to participants by giving them an opportunity to look at different development or maintenance methodologies and approaches.

Participants:

Each participant in the structured walkthrough process has a specific role. For the small size project, a person may fulfill multiple roles.

The **author** of the deliverable is responsible for requesting the walkthrough when a meaningful portion of the product has been developed and is free from casual errors (e.g., spelling errors). The author attends the walkthrough as observer and answers reviewer's general questions. The author is not a reviewer.

The **presenter** usually develops the agenda for the walkthrough and presents the deliverable being reviewed. The presenter should be familiar with the deliverable and be a member of the project team.

The **moderator** facilitates the walkthrough session, ensures that the walkthrough agenda is followed, and encourages the participation of all reviewers. The moderator may also be the scribe.

The **reviewers** evaluate the deliverable to determine if it is technically accurate. The reviewers also assess whether the project guidelines or standards are being followed, the project requirements are met, and the product is properly prepared.

The **scribe** takes notes during the walkthrough. The scribe records the errors identified and any other technical comments, suggestions, and unresolved questions. The scribe should not be a reviewer.

Meeting Record:
The Structured Walkthrough Meeting Record worksheet is available to assist the reviewers with recording errors found prior to the walkthrough session, and for the scribe to record information discussed during the walkthrough. The worksheet is divided into two parts: Part 1 is used to record administrative meeting information; Part 2 is used to record reviewer comments, questions, and follow-up action items. A template of the worksheet is provided as an attachment to this procedure.

Implementation:
This procedure describes a formal structure for conducting walkthroughs. The formality and structure of the walkthrough sessions should be tailored to meet the needs of the development or maintenance team, and the purpose and scope of the deliverable.

Note: The Structured Walkthrough procedure is a Key Process Area at Level 3 of the SEI Software Capability Maturity Model.

Responsibilities Before the Walkthrough

Author's Responsibilities:

The author of the deliverable is responsible for the following activities prior to the walkthrough session.

STEP	ACTIVITY
1	Complete a meaningful segment of a deliverable. Avoid requesting a walkthrough on an incomplete segment or on a segment(s) that is too large to be adequately reviewed in less than 2 hours.
2	Proofread deliverable segment to eliminate non-technical errors such as spelling or typographical mistakes. Non-technical errors can distract reviewers from the technical aspects of the deliverable.
3	Notify the presenter that a completed segment of a work product is ready for a structured walkthrough. The author may discuss potential reviewers with the presenter.
4	Prepare any support materials (such as flow charts) to assist reviewers with their understanding of the entire deliverable and how the segment being reviewed fits into the entire product.
5	Provide the deliverable and all support materials to the presenter for advance distribution to the reviewers.
6	When the segment to be reviewed is finished, the author should be prepared to work on other segments of the deliverable (or other project tasks) while waiting for the walkthrough to occur.

Presenter's Responsibilities:

The presenter is responsible for the following activities prior to the walkthrough session.

STEP	ACTIVITY
1	Determine if the size of the deliverable segment is appropriate for one walkthrough session. The duration of a walkthrough session should not exceed 2 hours. If more time is necessary, the deliverable segment should be divided into smaller portions and each portion reviewed separately.
2	Select reviewers who are appropriate for the deliverable; such as systems analysts, programmers, technical writers, and testers. Reviewers should include people on and off the project. In some cases, the participation of software customers may be considered desirable. If necessary, the presenter can discuss who should participate in the walkthrough with the manager of the project team.
3	Select the moderator and the scribe. Determine whether the scribe will be responsible for completing the Structured Walkthrough Management Summary Report.
4	Schedule the meeting date, time, and location. Notify all participants of these arrangements at least 2 days prior to the walkthrough.
5	Establish the agenda. Review the agenda and any important issues with the moderator.
6	Provide reviewers with copies of all materials to be reviewed at least 2 days prior to the walkthrough. The review package should include a blank copy of the Structured Walkthrough Meeting Record worksheet for optional use by reviewers.

Responsibilities Before the Walkthrough

Reviewers' Responsibilities:

The reviewers are responsible for the following activities prior to the walkthrough session.

STEP	ACTIVITY
1	Carefully review the materials provided by the presenter. Make a note about the amount of time spent reviewing the material. Give this information to the scribe at the beginning of the walkthrough session.
2	2 Identify technical errors. Insert comments and questions directly on the review materials or on part 2 of the Structured Walkthrough Meeting Record worksheet for easy reference during the walkthrough discussion.
3	Note directly on the review materials any non-technical errors found during the review, such as spelling or typographical mistakes. While these errors are not discussed during the walkthrough, they should be provided to the author at the conclusion of the walkthrough.
4	Notify the presenter immediately if the reviewer will not be able to complete the review in time for the walkthrough session. An unprepared reviewer will hinder the walkthrough process. If enough time is available, the presenter can select a new reviewer.
5	Review the procedures for the structured walkthrough process. Each reviewer should be familiar with the procedures prior to participating in a walkthrough session

Moderator and Scribe Responsibilities:

The moderator and scribe are responsible for the following activities prior to the walkthrough session.

STEP	ACTIVITY
1	Review the materials provided by the presenter to become familiar with the contents.
2	Review the agenda and discuss any questions with the presenter.
3	Note directly on the review materials any non-technical errors found during the review, such as spelling or typographical mistakes. While these errors are not discussed during the walkthrough, they should be provided to the author at the conclusion of the walkthrough.
4	Review the procedures (ground rules) for the structured walkthrough process. Clarify specific roles and responsibilities with the presenter. The moderator and scribe should be familiar with the procedures prior to participating in a walkthrough session.

Structured Walkthroughs for Lifecycle Phases

Moderator's Responsibilities:

The moderator is responsible for the following activities during the walkthrough session.

STEP	ACTIVITY
1	Call the walkthrough session to order. It is important to start the session at the scheduled time.
2	Ask participants to introduce themselves and state their current responsibility/job assignment.
3	Briefly review the procedures and agenda for the walkthrough session.
4	Facilitate the walkthrough session. Every attempt should be made to adhere to the agenda and the established meeting procedures. Encourage active participation of all reviewers. Limit discussion to the identification of errors. The discussion of solutions is not part of the walkthrough process. Limit the author's participation to observation and answering questions. If the session exceeds 2 hours, the moderator should stop the session at a logical breaking point and schedule another session to continue the discussion. When walkthrough sessions exceed 2 hours, the productivity and attention span of the reviewers will be adversely affected.
5	At the conclusion of the session, ask the reviewers to make a decision about the status of the deliverable as follows: A Accept product as is B Revise--no further walkthroughs for this product C Revise and schedule another walkthrough A majority opinion decides the action. If a majority opinion or consensus cannot be reached, the presenter will make the decision. If another walkthrough is necessary, the entire structured walkthrough process should be repeated.
6	Adjourn the walkthrough session at the scheduled time. If the agenda has not been completed, schedule a follow-up session.

Structured Walkthroughs for Lifecycle Phases

Presenter's Responsibilities:

The presenter is responsible for the following activities during the walkthrough session.

STEP	ACTIVITY
1	Provide a brief overview of the deliverable.
2	If necessary, review outstanding issues from previous walkthrough(s).
3	Present the product to be reviewed. Answer reviewers' questions. The presenter can ask the author for assistance in answering questions.
4	At the conclusion of the meeting, if the reviewers cannot reach consensus about the status of the deliverable, the presenter is responsible for making that decision. The status will be one of the following: A Accept product as is B Revise--no further walkthroughs for this product C Revise and schedule another walkthrough

Scribe's Responsibilities:

The scribe is responsible for the following activities during the walkthrough session.

STEP	ACTIVITY
1	Record the beginning time for the walkthrough session.
2	Record the amount of time each reviewer spent reviewing the deliverable.
3	Record the attendance of each participant.
4	Record the technical errors identified by the reviewers. Record all significant comments and suggestions made by the reviewers and presenter.
5	Record suggested action items and other follow-up activities.
6	Record the end time for the walkthrough session.

Structured Walkthroughs for Lifecycle Phases

Reviewers' Responsibilities:

Each reviewer is responsible for the following activities during the walkthrough session.

STEP	ACTIVITY
1	Provide the scribe with the time spent reviewing the work product.
2	Provide the appropriate introduction information (e.g., name and current job responsibilities).
3	Describe technical errors found during review of the work product. Be an active participant.
4	Ask questions as needed to clarify information about the work product.
5	Make constructive suggestions and comments about the work product.
6	Participate in the decision about the status of the deliverable: A Accept product as is B Revise--no further walkthroughs for this product C Revise and schedule another walkthrough If consensus cannot be reached by the reviewers, the presenter is responsible for making the decision.
7	Inform the author about any non-technical errors found during the review by providing a marked up copy of the review package.

Scribe's Responsibilities:

The scribe is responsible for the following activities after the walkthrough has taken place.

STEP	ACTIVITY
1	Prepare the meeting record for the walkthrough session. Include any action items identified by the reviewers and the person/team responsible for completing each action item.
2	Circulate the meeting record to the participants for their review and comments.
3	Update the meeting record as needed. Distribute the revised meeting record to the author. Copies of the meeting record should be distributed to the other participants only if an additional walkthrough is required.

Reviewers' Responsibilities:

The reviewers are responsible for the following activities after the walkthrough.

STEP	ACTIVITY
1	Review the walkthrough session meeting record for accuracy and completeness.
2	Indicate changes that are needed to add or clarify information in the meeting record. Submit any changes to the scribe. If necessary, discuss discrepancies with the presenter.
3	If requested by the author of the deliverable, provide additional explanation of walkthrough comments.

Structured Walkthroughs for Lifecycle Phases

Presenter's Responsibilities:

The presenter is responsible for the following activities after the walkthrough.

STEP	ACTIVITY
1	Review the walkthrough session meeting record for accuracy and completeness.
2	2 Indicate changes to the meeting record and return to scribe. If necessary, discuss discrepancies with the reviewers.
3	Initiate follow-up activities recommended by the reviewers. Verify that all action items have been assigned to the appropriate person/team.
4	Complete a Structured Walkthrough Management Summary Report. Include the following information: • Description of the deliverable reviewed. • Description of findings. In addition to findings, include • significant problems that would cause schedule slippage • or project cost increase. • Date, time, and duration of the walkthrough. • List of attendees. • Status decision (i.e.; accept as is, revise--no further walkthrough, or revise and schedule another walkthrough) and any other follow-up activities.
5	Distribute copies of the Structured Walkthrough Management Summary Report to the appropriate management personnel including the Project Manager and the Quality Assurance Team Manager.
6	Track progress made on open action items. As action items are closed, indicate closed status on the meeting record.
7	If necessary, schedule a follow-up walkthrough when the revised deliverable is ready.

Structured Walkthroughs for Lifecycle Phases

Author's Responsibilities:

The author is responsible for the following activities after the walkthrough.

STEP	ACTIVITY
1	Make all necessary changes to the deliverable.
2	Use the structured walkthrough meeting record as a checklist to make sure all errors are corrected, reviewers comments have been addressed, and open issues are investigated.
3	Check with the presenter and reviewers, as needed, to obtain additional information or clarifications.
4	Conduct follow-up meetings with subject matter experts, as needed, to complete deliverable.
5	Prepare deliverable and participate in follow-up walkthrough, if required.

Quality Assurance Team Manager:

The Quality Assurance Team Manager is responsible for the following activities after the walkthrough.

STEP	ACTIVITY
1	Prepare a summary of the information contained in the Structured Walkthrough Management Summary Report.
2	Distribute the summary to the Technical Monitor for the software development task. The data presented in the report is included in monthly management reports.

Project Manager*:* The Project Manager is responsible for the following activities after the walkthrough.

STEP	ACTIVITY
1	Review the Structured Walkthrough Management Summary Report.
2	If a problem exists that would cause a schedule slippage or project cost increase, send written notification to the project's Technical Monitor. An electronic mail message is an acceptable form of notification.
3	File the Structured Walkthrough Management Summary Report in the project management notebook/files.
4	Follow up on any action items that remain open. A formal plan may need to be developed for action items that cannot be resolved during the current lifecycle phase.

Structured Walkthroughs for Lifecycle Phases

Project Technical Monitor:

The Project Technical Monitor is responsible for the following activities after the walkthrough.

STEP	ACTIVITY
1	Review the summary information about the completed structured walkthroughs that is provided by the Quality Assurance Team Manager.
2	The Technical Monitor reports walkthrough statistics to Appropriate levels of Agency senior management.

Preparation of Summary Report:

The presenter is responsible for the preparation of the Structured Walkthrough Management Summary Report (Summary Report). The presenter may ask the scribe to prepare the report. If the scribe prepares the report, the presenter reviews the report before it is distributed. A sample Summary Report is provided at the end of this appendix.

The Summary Report is distributed to the appropriate project personnel including:

- Project Manager
- Quality Assurance Team Manager

The Summary Report is used by the Quality Assurance Team to maintain statistical data on structured walkthroughs.

The Summary Reports generated during each phase of the software lifecycle will be checked during the In-Phase Assessments (IPAs). The purpose of the IPA check is to verify that structured walkthroughs were conducted during each lifecycle phase, that the walkthrough action items were documented, and that the action items have been properly resolved and closed.

Follow-up Walkthrough:

If a follow-up walkthrough is required, the procedures used in the original walkthrough should be repeated. Use the meeting record from the previous walkthrough as a checklist to confirm that the previously identified errors and issues were resolved.

Structured Walkthrough Meeting Information

Introduction: Structured walkthroughs are generally used to review software products or systems under development or maintenance at various lifecycle phases. This section describes the deliverables that should be reviewed at each phase of the lifecycle. The deliverables correspond to the deliverables described in the *State of Michigan's Systems Development Lifecycle.*

Planning Phase:

The Planning Phase defines the work to be accomplished for a development or maintenance tasks and estimates the resources that will be required. During the Planning Phase, a structured walkthrough should be conducted for each project management tool.

Purpose	Participants
Reviews the phase deliverables, such as the following: ❑ Feasibility Study/Statement ❑ Project Plan ❑ Deliverables log ❑ Milestone schedule ❑ Work Breakdown Structure ❑ Software Quality Assurance Plan ❑ Configuration Management Plan	The developer and at least one systems analyst, preferably outside the project. If the project involves telecommunications (e.g., LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

Requirements Definition Phase:

The Requirements Definition Phase determines the scope and requirements for a development or maintenance project. During the Requirements Definition Phase, structured walkthroughs are used to identify problems, inaccuracies, ambiguities, and omissions in the Requirements Specifications.

Purpose	Participants
Reviews the following phase deliverables: ❑ Requirements Specification - Client goals - Data requirements - Functional requirements - Performance requirements • Interface requirements • Communications • Computer Security ❑ Continuity of Operations Plan ❑ Project Test Plan ❑ Acceptance Test Plan	One or more of the project designers and at least one systems analyst. If the project involves communications (e.g., - telecommunications; LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

Structured Walkthrough Meeting Information

Functional Design Phase:

The Functional Design Phase selects the design elements that determine how the software product will be constructed to meet the functional requirements. During the Functional Design Phase, the structured walkthroughs are used to identify flaws, weaknesses, errors, and omissions in the architecture of the design.

Purpose	Participants
Reviews the Functional Design Document, logical model, data dictionary, and requirements traceability matrix for errors in the following design areas: ❑ Hardware ❑ Software ❑ Logical design ❑ Communications ❑ System interfaces ❑ Backup and recovery ❑ Security ❑ Customer interface ❑ Reports	At least one systems analyst and one or more of the project designers/programmers. If the project involves telecommunications (e.g., - LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

System Design Phase:

The System Design Phase uses the concepts and the system architecture to describe the system components in detail. During the System Design Phase, structured walkthroughs are used to review detailed specifications, and plans that address testing and implementation issues.

Purpose	Participants
Reviews the following phase deliverables: ❑ Physical Model ❑ Programming Standards ❑ Program Specifications ❑ System Design Document ❑ Conversion Plan ❑ Integration Test Plan ❑ System Test Plan	At least one systems analyst and one or more of the project designers/programmers. If the project involves telecommunications (e.g., - LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

Structured Walkthrough Meeting Information

Programming Phase:

The Programming Phase involves the construction of the software product and the testing that is an integral part of the construction process. During the Programming Phase, walkthroughs are conducted on clean compiles, test data bases, and the operating documentation.

Purpose	Participants
Reviews listing (clean compiles) for approximately 80 hours of coding, or at the completion of a logical unit of work. Reviews should verify adherence to the following: ❑ System design ❑ Programming standards ❑ Program Specifications ❑ Software Configuration Management Plan _____ Determines the adequacy of the Integration and System test data base and test plans.	Technical personnel with appropriate expertise and at least two additional reviewers. The entire programming team might attend the walkthrough, depending On the approach. If the project involves telecommunications (e.g., - LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

Software Integration and Testing Phase:

The Software Integration and Testing Phase is the transition from individual software components to an integrated software product. During the Software Integration and Testing Phase, structured walkthroughs are used to test the integrated product, check the accuracy of the operating documents that will be provided to the customer(s) and maintenance programmer(s), and plan for the acceptance activities.

Purpose	Participants
Reviews the following phase deliverables: ❑ Integration and System Test ❑ Results/Reports ❑ Procedure Manual ❑ Programmers Reference ❑ Manual ❑ Installation Plan ❑ Acceptance Test Plan ❑ Training Plan ❑ Pre-acceptance Checklists ❑ Maintenance Plan Reviews the following production activities: ❑ Data conversion ❑ Installation procedures	Participants include personnel with appropriate technical expertise and a technical writer. If the software product is an application on a mainframe platform, include a representative from appropriate area. If the project involves telecommunications (e.g., - LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

Structured Walkthrough Meeting Information

Installation and Acceptance Phase:

The Installation and Acceptance Phase is the transition from a software product in development to a product or system in full production status. During the Installation and Acceptance Phase, structured walkthroughs are used to check the Acceptance Test Report and inspect the plans for activities performed in preparation for full-scale production.

Purpose	Participants
Reviews the following phase work with products: ❑ Acceptance Test ❑ Results/Report ❑ Acceptance Checklist Performs the following production activities: ❑ Data conversion ❑ Installation procedures	Participants include personnel with Appropriate technical expertise. If the software product is an application on a mainframe platform, include a representative from appropriate area. If the project involves telecommunications (e.g., LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

Types of Documents:

Structured walkthroughs are appropriate for reviewing other types of documents, such as the following:

- ❑ Departmental and contractual publications
- ❑ Long-range plans
- ❑ Administrative and technical operating procedures
- ❑ Technical reports
- ❑ Presentations

Structured Walkthrough Meeting Information

Types of Verification:

When reviewing other types of documents, structured walkthroughs are used to verify the technical and editorial accuracy and appropriateness of the content and format.

Purpose	Participants
Reviews for accuracy including the following: ❑ Consistency ❑ Completeness ❑ Conformance to standards ❑ and guidelines ❑ Style ❑ Grammar and spelling	Technical experts, technical writer, and graphics expert.

Structured Walkthrough Meeting Information

Part 1: Structured Walkthrough Meeting Information

Complete the following blocks to record the structured walkthrough meeting information. If you have any questions, call your Quality Assurance representative.

Walkthrough Information	Participant Information		
	Participant Name/ Responsibilities	**Preparation Time**	**Present (✔)**
Project name:	Author:		
Walkthrough date:	Presenter:		
	Moderator:		
	Scribe:		
Walkthrough deliverable Decision: ❑ Acceptable as presented ❑ Acceptable with revisions ❑ Revisions with another walkthrough	Reviewer: Reviewer: Reviewer: Reviewer:		
If a follow-up walkthrough is required: Date: Location: Time:	Reviewer:		

Part 2: Structured Walkthrough Findings

Record reviewer comments and action items identified during the structured walkthrough in the blocks below. If more space is needed, make additional copies of this page.

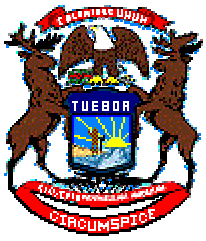
Number	Comments and Action Items	Date Closed

Structured Walkthrough Meeting Information

Structured Walkthrough
Management Summary Report

Product Name:	
Product/Module Number:	Acronym:
Deliverable (check one)	
__Feasibility Study/Statement __Project Plan __Software Quality Assurance Plan __Software Configuration Management Plan __Continuity of Operations Plan/Statement __Data Dictionary __Requirements Traceability Matrix __Software Requirements Specification __Project Test Plan __Acceptance Test Plan __Design Methodology __Logical Model __Functional Design __Physical Model Other (specify):	__Integration Test Plan __System Test Plan __Conversion Plan __System Design Document __Program Specifications __Programming Standards __Acquisition Plan __Installation Plan __Source Code __Transition Plan __Procedure Manual __Programmers Reference Manual __Document Training Plan __Other (specify):

Description of Product Reviewed: Summary of Findings: (Include significant problems that would cause schedule slippage or project cost increase)
Date: Start Time: End Time: Duration:
Reviewers: Presenter: _____ _____ Moderator:_____ _____ Scribe:_____ _____
Decision: A = Accept product as is B = Revise--no further walkthrough for this product C = Revise and schedule another walkthrough Enter appropriate letter:



SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX D

IN-PHASE ASSESSMENT PROCESS GUIDE

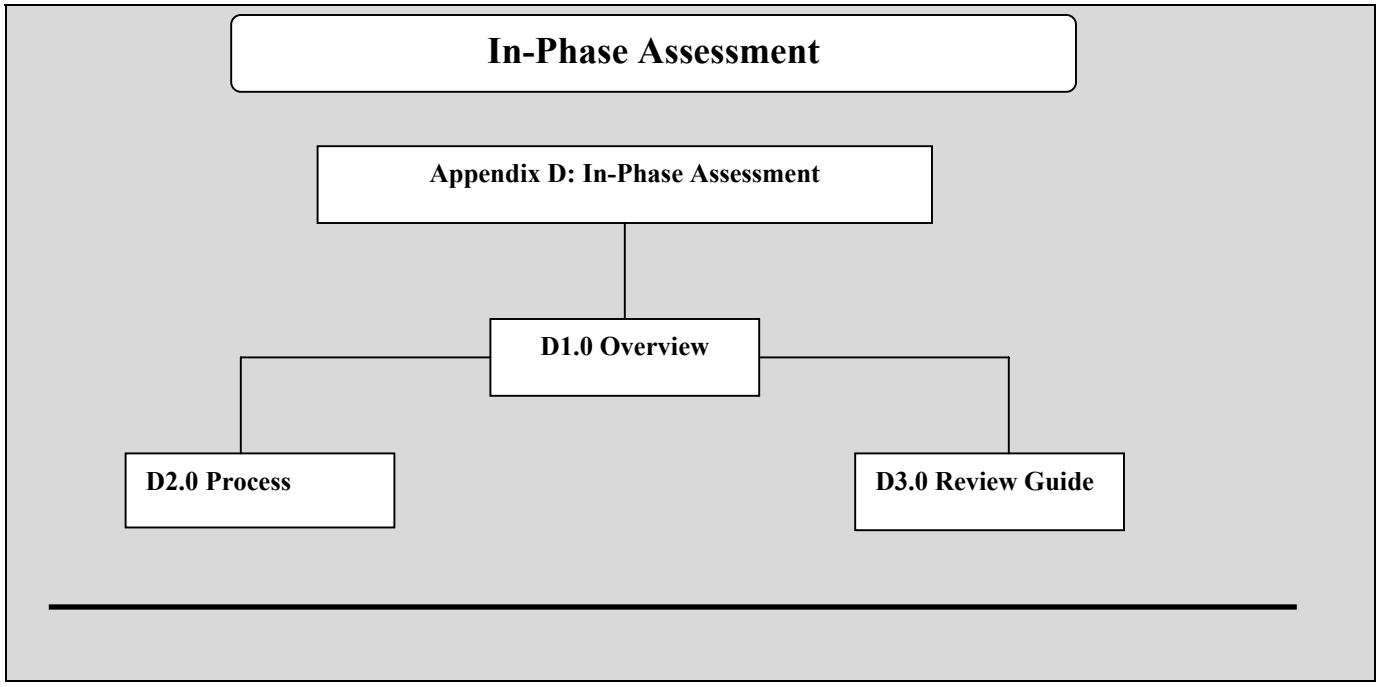
Appendix D: In-Phase Assessment

Table of Contents

Table of Contents

Appendix D: In-Phase Assessment

Highlights of Phase



Appendix D: In-Phase Assessment

Section D1.0 Overview

Introduction:	In-Phase Assessments (IPAs) are independent reviews of agency system development and maintenance projects. IPAs are conducted in all phases of the software development and maintenance lifecycle (SDLC) in accordance with the IPA schedule, which is documented in the project plan. This document defines the process for planning and conducting IPAs. An In-Phase Assessment is similar to a Software Quality Assurance Review or an Independent Verification and Validation process.
Purpose:	The purpose of IPAs is to assure, via an independent assessment, that the established system development and project management processes and procedures are being followed effectively, and that exposures and risks to the current project plan are identified and addressed. An IPA is a project review that is conducted by a reviewer independent of the project. The reviewer assesses a project's processes, and deliverables to verify adherence to standards and that sound system development and project management practices are being followed. An IPA is a paper review and agency does not require meetings among the involved parties.
Who Conducts:	Within the current framework of deployment of this process, the Quality Assurance Analyst will conduct the IPAs.
Applicability:	This process is applicable to all SOM development and maintenance efforts that would follow the SOM Systems Development Lifecycle (SDLC) . [1]
Timing/Frequency:	An IPA can be conducted anytime during a phase whenever a deliverable is stable enough, or near the end of a phase to prepare for phase exit.

[1] The SDLC is the standard for all SOM projects.

Appendix D: In-Phase Assessment

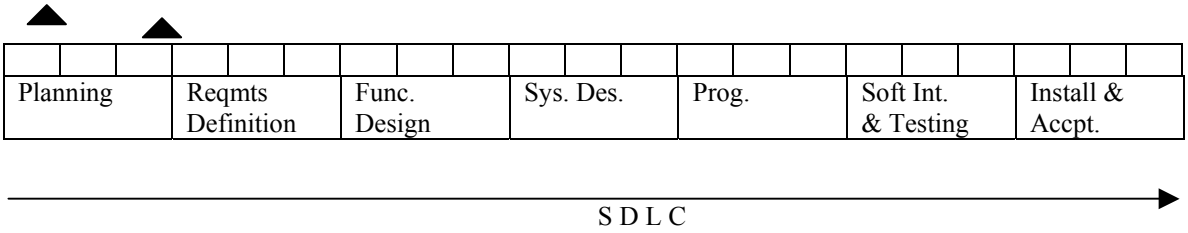
Section D1.0 Overview

Diagram:

The following diagram shows the timing of IPAs relative to the SDLC lifecycle. The break out of the IPAs shown in the planning phase also would apply to the other SDLC phases.

IPA

Phase exit (this is a separate process)



In-Phase Assessment:	One or more in-phase IPAs can be scheduled for a given phase of development. The purpose of this review is to assess one or more deliverables when development (of that deliverable) is far enough along to allow for review, and early enough to allow for revisions prior to phase exit. The results of the review are contained in a report that is submitted directly to the project manager.
End of In-Phase Assesssment:	An end-of-phase IPA must be conducted near the end of each phase of development. The purpose of this review is to assess the readiness of a project to proceed to the next phase by reviewing all the deliverables for the current phase. The results of this review are contained in a report that is submitted to the project manager. Copies of the report may be provided to the next-level manager and the client or system owner, as appropriate. In order to exit the current phase of development, the project manager must develop an acceptable action plan to address any open issues or qualifications.
Process Ownership:	The responsibilities of the In-Phase Assessment process sponsor include approving the initial process definition document and changes during process improvement. The responsibilities of the process owner include initial process definition, process implementation, and conducting ongoing process improvement. The process was originally developed with the support of a cross-functional process team.
Change Control:	The IPA process is a component of the SDLC. Changes to this process will be instituted using the same change mechanism that has been implemented to process changes to the SDLC.
Relationship to Other System Development Processes:	The IPA process is a primary component of the SOM SDLC. Together with other processes it serves to assure a consistent and predictable outcome in the resulting software products. The IPA process is complementary to other processes such as Phase Exits and Structured Walkthroughs.

Section D1.0 Overview

Process Measurements:

Process measurements are required in order to understand, in a quantifiable manner, the effectiveness of a process at work. If measurements indicate the process is not working as designed, a causal analysis should be conducted to identify the root cause of the breakdown, and changes should be implemented via the process improvement team. The IPA process is considered to be effective (working as designed) if:

- All issues that must be resolved in the current phase are identified.
- Unmet project objectives can be attributed to issues documented in an IPA.
- All issues without an acceptable action plan become qualifications to exit the current phase of development.
- All issues are properly documented.
- Specific procedures for capturing the data for the above measurements will be defined during process improvement.

Section D2.0 Process

Scope:

The In-Phase Assessment process begins with the scheduling of the review and ends with the delivery of the report developed after the review.

It is the responsibility of the project manager to develop and implement solutions for the issues and qualifications documented during the review. The project manager must develop an appropriate action plan for each issue.

Customers:

The customers of the IPA process are those individuals or organizations that will use the output of the IPA process. The primary customers of the process are:

- Project manager
- Project manager's manager
- Quality Assurance (QA)

Secondary customers of the process are:

- System owner
- Customer POC
- Client representative

Suppliers:

The project team develops deliverables that become input to the IPA process.

Input:

The following are the minimum inputs to the IPA process:

- Software development lifecycle deliverable(s)
- Project plan developed during the planning phase, which includes the work breakdown structure and timeline in addition to other components
- Updated project plan revised during all subsequent phases
- Structured walkthrough minutes/conference records

Section D2.0 Process

Diagram:

The following diagram depicts the IPA process flow:

Planning Phase:

PLAN FOR THE REVIEW	IPA target dates defined in the project plan for each phase

All Phases:

SCHEDULE THE REVIEW	Set date Secure agreement(s)
RECEIVE DELIVERABLES	Phase deliverables Updated project plan Walkthrough minutes Other applicable items
CONDUCT THE REVIEW	Review project plan Review other items Review deliverables Formulate assessment
PREPARE REVIEW FINDINGS	Assessment of risk to achieve plan List of concerns Recommendations

Discuss review findings with project manager.

Section D2.0 Process

Planning In-Phase Assessments:

In the planning phase, the target date for conducting the IPA(s) at each phase of development is documented in the project plan. In-phase IPAs can be conducted at any logical point in the phase. End-of-phase IPAs should be scheduled near the end of a phase (e.g., 2 or 3 weeks ahead of the phase exit milestone). In-phase IPAs should be scheduled for the next phase only, since reviews for all subsequent phases might be difficult to plan in advance.

Schedule Review:

In each phase, as soon as practical, the actual assessment point should be established and agreed-to, and the activity scheduled so that the project manager is aware and there are no surprises.

Receive Deliverables:

The reviewer should be provided with a copy of the deliverables to be reviewed, plus the current project plan if it is an end-of-phase IPA. The deliverables will vary according to the project's phase of development. The System Development Review Level table in section D3.2 provides detailed review guidance.

Conduct Review:

The reviewer should examine each lifecycle deliverable. The depth of the examination will vary according to the deliverable and the project's phase of development. Guidance to assist the reviewer is provided in Section D3.0, Review Guide.

Review Levels:

The following are the levels of review that can be performed on the SDLC deliverables. For each deliverable, specific guidance is provided in Section D3.0, Review Guide.

Level Explanation:

1 Verify the existence of the deliverable.
Review to assure the deliverable exists and is complete.

2 Verify minimum content exists.
Review to ensure the minimum level of information has been provided. Verify the existence of content by checking sections/headings.

3 Verify content is rational.
Review to make judgements as to the quality and validity of the deliverable.

Prepare Review Findings:

The reviewer should document the results of the IPA and develop the report described in the Process Output section.

Section D2.0 Process

Process Output:

A report must be developed when the IPA process is executed. Each IPA requires a report even if no issues were identified during the review. The report should be brief with the focus on providing a clear statement of the issues(s); solutions may be suggested, but are the project manager's responsibility. The report should include the following elements:

- A written assessment of the current project plan in terms of the following:
- Risk to schedule and budget
- Risk for next phase
- Risk for remainder of project

Risk categories:

- **Low** - Potential or existing problems must be addressed to avoid an impact to the current project plan. This would also apply if no issues were identified.

- **Medium** - Problems exist that have a high probability of impacting the current project plan or other dependencies.

- **High** - Serious problems exist (without an acceptable plan to resolve) that have a high probability of impacting customer acceptance, the current project plan, or other dependencies.

A list of issues/concerns if any were formed during the review. An issue is logged if there is a problem without a visible plan for resolution. Once a list of issues have been compiled, it should be reviewed with the project manager to see if any new or additional information might mitigate or eliminate any of them. Remaining issues must be addressed with an action plan from the project manager. Issues from an end-of-phase IPA might become "qualifications" to exiting the current phase of development. Refer to the phase exit process documentation for additional information.

The following are some examples of issues.

- No description of the estimating methodology used.
- No definition of a change control mechanism.
- Signoff (concurrence) from the prior phase is not visible.
- Concern about the appropriateness of the process used to arrive at technical decisions. In this example, the reviewer may recommend an additional in-depth review by technical experts as an action item.

If no issues were identified, the report only needs to contain the name of the project, date of the review, reviewer name, and a statement that no issues were identified.

Process Output Continued:

Additional reviewer comments, as appropriate. These include suggestions and recommendations that would benefit the project. The reviewer is encouraged to provide this feedback based on his/her experience. Reviewer comments are provided for the benefit of the project manager and should not be logged as issues requiring an action plan. In certain cases the reviewer may also recommend a more in-depth review by an individual highly skilled in a certain area, to help validate technical decisions and conclusions.

For In-Phase IPAs, the written report is distributed to the following:

- Project manager

For end-of-phase IPAs, the written report is distributed to the following:

Project manager, with copies to:

- Project manager's manager
- Quality Assurance manager

In-Phase Assessment (IPA) Report

State of Michigan
(Insert Agency Name Here)
In-Phase Assessment Report

A. General Information

Information to be provided in this section gives a specific name to the project as well as pertinent information about the personnel involved.

Project Name: _____ ***Date:*** _____

Controlling Agency: _____ ***Modification Date:*** _____

Phase: _____ ***Phone:*** _____

Prepared by: _____ ***Authorized by:*** _____

B. In-Phase Assessment Issues

Describe the In-Phase Assessment issues and concerns. List how the issues or concern was resolved.

Number	Issues/Concerns	Resolved

C. In-Phase Assessment Corrective Action

Describe the In-Phase Assessment correction actions and recommendations.

Number	Corrective Action	Recommendations

Assessment of risk to schedule:

	Low	Medium	High
For the next phase			
For the remainder of the project			

Responsibility Matrix

The following matrix defines the responsibilities for the various organizations involved in the IPA process:

	Project Team	Quality Assurance	IPA Reviewer (1)
Prepare deliverables	P		
Schedule IPA	P	R	
Conduct IPA	S	P	
Compile list of issues	R	P	
Prepare assessment		P	
Ensure issue resolution	P		
Monitor process effectiveness		P	
Continuous process improvement	S	P	

Legend

P = Perform R = Review S = Support

(1) As the process is currently implemented, the IPA is conducted by Quality Assurance. However, it is possible for the IPA to be conducted by other parties, e.g. a peer project manager. Section D3.1, Reviewer Selection, provides a description of the skills required to conduct IPAs.

Section D3.0 Review Guide

Introduction:

This section contains guidance on selecting an IPA reviewer, the appropriate level of review to be performed for each deliverable, and what to look for in each. While some deliverables are prepared once during the applicable phase of system development or maintenance, others are subsequently updated as the project progresses. These variances are highlighted in the review level tables.

Guidance:

The following guidance is provided in this section:

D3.1 Reviewer Selection
D3.2 Deliverables Review Guidance
D3.3 Project Management Review Guidance

D3.1 Reviewer Selection:

The review must be conducted by a different work group not support the same organizational unit as the development or maintenance team. This allows for an independent view of problems and issues that might exist and serves as a cross-training tool. The experience and skills required include:

- Hands-on experience planning and managing technically complex software development projects.
- Working knowledge of the SOM Systems Development Lifecycle (SDLC).
- Ability to deal with people and communicate well.

Individuals who typically have the technical background, experience, and skills required include team managers, area managers, project managers, project leaders, task leaders, quality assurance representatives.

D3.2 SDLC Deliverables Review Guidance:

This section provides guidance on what to look for when reviewing and assessing the various SDLC deliverables and components of the project plan. Deliverables are developed throughout the SDLC. They serve to document all project-related data and form the basis of understanding between all parties involved in developing systems. These deliverables are required at various phases of the SDLC.

For a detailed description of the deliverables refer to the SOM SDLC.

Section D3.0 Review Guide

Guidance:

The review guidance for the following deliverables is provided in this section:

D3.2.1 Project Plan
D3.2.2 Structured Walkthrough Meeting Record
D3.2.3 Feasibility Statement
D3.2.4 Software Quality Assurance Plan
D3.2.5 Configuration Management Plan
D3.2.6 Requirements Specification
D3.2.7 Project Test Plan
D3.2.8 Continuity of Operations
D3.2.9 Acceptance Test Plan (draft)
D3.2.10 Logical Model
D3.2.11 Data Dictionary
D3.2.12 Cross Reference Matrix
D3.2.13 Preliminary Design Document
D3.2.14 Program Specifications
D3.2.15 Physical Model
D3.2.16 Integration Test Plan
D3.2.17 Conversion Plan
D3.2.18 System Design Document
D3.2.19 System Test Plan
D3.2.20 Software Baseline
D3.2.21 Acquisition Plan
D3.2.22 Transition Plan
D3.2.23 Procedure Manual
D3.2.24 Operating Documentation
D3.2.25 Test Reports
D3.2.26 Training Plan
D3.2.27 Pre-acceptance Checklist
D3.2.28 Installation Plan
D3.2.29 Acceptance Test Plan (final)
D3.2.30 Operational System
D3.2.31 Acceptance Test Report
D3.2.32 Maintenance Plan

The table on the following page identifies the appropriate level of review to be performed depending on the deliverable and the phase of development.

System Development Deliverables Review Level Table

		SDLC Phase						
Table Legend	**SDLC Deliverable**	**Planning**	**Requirements Definition**	**Functional Design**	**System Design**	**Programming**	**Software Integration & Test**	**Installation & Acceptance**
Review Levels: ❑ 1 -Verify the existence of the deliverable ❑ 2-Verify minimum content exists ❑ 3-Verify content is rational (1) See Project Management Review table for details	Project Plan (1)	3	3	3	3	3	3	3
	Walkthrough Records	2	2	2	2	2	2	2
	Quality Assurance Plan	3						
	Feasibility Statement	2						
	Configuration Mgmt. Plan	2						
	Requirements Specification		3					
	Project Test Plan		3					
	Acceptance Test Plan (draft)		2					
	Continuity of Ops. Plan		2					
	Logical Model			2				
	Traceability Matrix			2				
	Data Dictionary			2				
	Functional Design Document			2				
	Program Specifications				2			
	Physical Model				2			
	System Design Doc.				2			
	Conversion Plan (as req.)				3			
	System Test Plan				3			
	Integration Test Plan				3			
	Software Baseline					1		
	Acquisition Plan					2		
	Transition Plan					2		

System Development Deliverables Review Level Table (continued)

		SDLC Phase						
Table Legend	**SDLC Deliverable**	**Planning**	**Requirements Definition**	**Functional Design**	**System Design**	**Programming**	**Software Integration & Test**	**Installation & Acceptance**
Review Levels: ❑ 1 -Verify the existence of the deliverable ❑ 2-Verify Minimum Content exists ❑ 3-Verify content is rational (1) See Project Management Review table for details	Procedure Manual					2		
	Integration Test Plan (final)					3		
	Installation Plan					3		
	System Test Plan (final)					3		
	Operating Documentation					2		
	Test Reports						2	
	Training Plan						2	
	Pre-Acceptance Checklist						2	
	Acceptance Test Plan (final)						3	
	Operating Documentation (final)						2	
	Installation Plan (final)						3	
	Operational System							1
	Acceptance Test Report							1
	Maintenance Plan							2

Section D3.0 Review Guide

D3.2.1 Project Plan:

Refer to Section D3.3, Project Management Review Guidance, for project plan guidance.

D3.2.2 Structured Walkthrough Meeting Record (documentation) (level 2) :

The structured walkthrough meeting record (i.e. the documented results of the walkthrough), including the Management Summary and any other associated available documentation, should contain a list of any defects identified during that particular walkthrough, and a clear disposition for each defect. The reviewer's objective is to verify the following:

- ❑ One or more structured walkthroughs were conducted depending on the phase of development in accordance with SDLC guidelines.
- ❑ All defects have been addressed and closed; no action item has been left open-ended.
- ❑ If a follow-up structured walkthrough was required, the walkthrough was conducted and all action items were addressed.

Refer to Appendix C, Conducting Structured Walkthroughs, for additional information about structured walkthroughs.

D3.2.3 Feasibility Statement (level 2):

The feasibility statement is reviewed to determine if the following elements have been identified.:

- ❑ Project objectives
- ❑ System automation alternatives
- ❑ Potential technical solutions
- ❑ Benefits and costs
- ❑ Recommendations

The project file should be reviewed to verify that a structured walkthrough was conducted and any action items resulting from the walkthrough were addressed or resolved.

D3.2.4 Software Quality Assurance Plan (level 3):

The Software Quality Assurance Plan should clearly define the project's quality assurance policies and procedures. The plan should address the following:

- ❑ When In-Phase assessments will be conducted and by whom.
- ❑ Applicability of published standards and procedures.
- ❑ Monitoring for application of applicable standards and procedures.
- ❑ Assurance of resolution of discrepancies.
- ❑ Assessment of project progress.
- ❑ Assuring the integrity of the software product.

D3.2.5 Configuration Management Plan (level 2):

Refer to Section D3.3, Project Management Review Guidance, for configuration management plan guidance.

Section D3.0 Review Guide

D3.2.6 Requirements Specification (level 3):

The requirements document is reviewed (use random sampling as appropriate) to assure that it includes the following elements:

- ❑ Project objectives are consistent with the objectives identified in the management plan.
- ❑ Requirements exhibit good attributes including clarity (no ambiguity), statement of a business problem or need (not the solution), and sufficient detail to allow for testing.
- ❑ All system requirements are addressed including data, functional, operational,
  security, communications, and implementation requirements.

D3.2.7 Project Test Plan (level 3):

This document is reviewed to assure that it includes the following elements. (System Test plan details should be provided in the System Design Phase):

- ❑ Approach to testing (e.g. "...system testing will be conducted by an independent group...")
- ❑ Purpose and scope of test efforts to be conducted
- ❑ List of items that are planned to be tested, and items that will not be tested
  - Rationale for not testing what is not going to be tested
- ❑ Who (organization) that will be responsible for conducting the testing
- ❑ Which levels of testing are planned to be conducted (e.g. unit, integration, system, function, acceptance)
- ❑ Physical location(s) where testing is planned to be conducted
- ❑ List of known requirements for conducting testing activities (e.g. hardware,
  software, skills, space, equipment)
- ❑ Who is expected to sign-off and approve the tests

D3.2.8 Continuity of Operations Plan/Statement (level 2):

The continuity of operations plan (for mission essential applications) or statement (for non-mission essential applications) is reviewed to determine if it includes the following elements:

- ❑ Requirements for continuity of operation, such as data backup, data recovery, and operation startup
- ❑ Plans for backup and recovery operations
- ❑ Training requirements and plans so that the required skills will be in place to execute backup and recovery operations

Section D3.0 Review Guide

D3.2.9 Acceptance Test Plan (draft) (level 2):	The acceptance test plan should include the following elements: ❑ The timeframe when the test(s) are being planned to be conducted. ❑ A list of organization(s) from which individuals will conduct the tests. At a minimum, this should contain positions and skills required. ❑ The test cases and scenarios that are planned to demonstrate that the requirements have been satisfied in the new system or application. ❑ A list of, or reference to, requirements that the system should satisfy. ❑ Required hardware, software, documentation, special environmental or operational requirements, and any other special considerations (e.g., travel).
D3.2.10 Logical Model (level 2):	The logical model is reviewed to ensure it contains the following elements: ❑ Description of the final sources and destination of data ❑ Description of the net flow of data across the system boundary ❑ Complete picture of the system processes, data flow, and data stores ❑ Clear connections between the various pieces of the model
D3.2.11 Data Dictionary (level 2):	The data dictionary (data model) is reviewed to ensure that data elements are documented in detail to include attributes, known constraints, input sources, output destinations, and known formats. The dictionary can include business rules processing statistics, and cross-referencing information. The reviewer should assure him/herself that techniques have been employed (e.g. structured walkthroughs with appropriate persons) to ensure all data elements have been identified up to this point.
D3.2.12 Traceability Matrix (level 2):	Each requirement must be traceable to one or more design entities. The matrix is examined to ensure that it allows for the design entities to be traced back to the project requirements, and for verification that all requirements are satisfied by the design.
D3.2.13 Functional Design Document (level 2):	The Functional Design Document is reviewed to assure that it describes the functions of the system in customer terminology. It should be written from the customer/system owner's perspective. It should enable the owner/customers to understand how the design will satisfy the requirements, providing an opportunity to give feedback before the design is completed. Under separate covers, or as sections of the Preliminary Design Document, the reviewer should ensure that the following design related deliverables have also been documented: ❑ The design alternatives including evaluation criteria, alternative descriptions, and recommendations. (Note: A reference to the alternatives study is acceptable).

Section D3.0 Review Guide

D3.2.13 Functional Design Document (level 2) Continued:

- ❑ The operating environment including hardware, software, communications, and interfaces.
- ❑ System design demonstrating the system architecture, system inputs, outputs, interfaces, and end-customer interfaces.
- ❑ Design methods used, design entities, and design dependencies.
- ❑ Security and control measures that will be incorporated into the software system.
- ❑ The display conventions that will be followed for the design of all end-customer interfaces (such as application screens).
- ❑ The implementation approach, reflecting the necessary planning to take the system through to implementation.

D3.2.14 Program Specifications (level 2):

This deliverable may consist of one or more documents. The specifications are reviewed to assure inclusion of the following elements:

- ❑ Program design characteristics including program architecture, and software and common program features.
- ❑ Standards and guidelines that will govern each program coded for the system.
- ❑ Program specifications (use sampling as appropriate - good rule of thumb is 15 percent). This should include a description of the logic and other particulars for each program in the system.

D3.2.15 Physical Model (level 2):

The physical model is reviewed to determine if it includes the following attributes and components:

- ❑ Contents and organization of the physical data files
- ❑ Processes and data flows among the files
- ❑ Modules or groups of modules structured into a hierarchy of sub-systems
- ❑ Manual tasks with physical input or output characteristics.

D3.2.16 Integration Test Plan (level 3):

This document (or section) is reviewed to assure that an incremental integration and test plan exists. It should include the following elements:

- ❑ Purpose and scope of test efforts to be conducted
- ❑ List of items to be tested, and items that will not be tested
  - Rationale for not testing what is not going to be tested
- ❑ Responsible individual (or organization) for each activity including sign-offs, management, and acceptance
- ❑ Schedules
- ❑ Description of "how" the testing will be conducted

Section D3.0 Review Guide

D3.2.17 Conversion Plan (level 2):	The conversion plan addresses the process of transitioning from the existing manual or automated process to the new application or system. This plan is reviewed to assure that it includes the following elements: ❑ General information. This section should describe, in a concise manner, relevant information about the system and task. ❑ Data conversion. This section should describe the strategy and specific activities required to convert data from the old system to the new one.
D3.2.18 System Design Document (level 2):	The System Design Document is reviewed to ascertain whether it translates requirements into precise specifications of the software components, interfaces, and data that are necessary before coding and testing can begin. The sequence and conditions inherent within modules should be documented. The requirement cross-reference matrix should be updated and should allow for each requirement to be traced to one or more of the detailed design entities to verify that all of the requirements will be satisfied by the detailed design.
D3.2.19 System Test Plan (level 3):	This document is reviewed to assure that it includes the following elements: ❑ Scope of the testing effort, and testing schedules. ❑ Objectives and definition of the test cases, and the hardware and software configuration for each test or set of tests. ❑ A requirement verification matrix mapping individual tests to specific requirements and specifying how each system requirement will be validated. ❑ Identification of test tools and test support needs (e.g. hardware / software to simulate production environment and conditions). ❑ Physical location(s) where testing is planned to be conducted. ❑ Who is expected to sign-off and approve the tests. ❑ How will fixes to defects be handled and how will re-testing be conducted.
D3.2.20 Software Baseline (level 1):	The reviewer should look for evidence that the software baseline (the system or application) has been completed. This may include a demo of the working application, a library listing showing all the coded programs, evidence of positive test results, and confirmation from the individuals who completed the testing activities.
D3.2.21 Acquisition Plan (level 2):	An acquisition plan should exist, if hardware, software, or services will need to be acquired at some point during the lifecycle in order to implement the system.

Section D3.0 Review Guide

D3.2.22 Transition Plan (level 2):

The Transition Plan should describe the detailed plans, procedures, and schedules that will guide the transition process to full operation of the system or application. It should demonstrate that it has been coordinated with operational and maintenance personnel. Refer to the SOM SDLC document, Section 8.7, Plan Transition to Operational Status for more details.

D3.2.23 Procedure Manual (level 2):

Note: The Procedure Manual may be developed as part of the Operating Documentation. The procedure manual (or user manual) is reviewed to determine if it contains sufficient information and detailed instructions required to access and use the system functions. For very small systems, a quick reference card may be more appropriate and sufficient. For larger systems, a reference card may also be provided in addition to the procedure manual. The extent of the procedure manual may also depend on the depth of the online help provided; the lower the level of online help the less extensive the procedure manual guide needs to be. A procedure manual should include:

- Overview of the system history, background, architecture, and current version
- Complete coverage of all system functions, in a logical order
- Samples of screens and reports, where appropriate to show examples
- Instructions for installing, configuring, and accessing the system
- Security features including what is accessible to each category of customers
- Who to contact for additional information or help

Additional note to reviewer: has the documentation team tested it, and have their comments been addressed?

D3.2.24 Operating Documentation (level 2) :

The system documentation should be complete, to allow for both ease of operation and maintenance. The following documentation should be developed:

- **Programmer's Reference Manual**. This document should include the information required to allow the programming staff to understand and maintain the system programs, databases, interfaces, and operating environments.
- **System Administration Manual.** This document should provide information necessary to enable the system administrator to understand and access the system functions required to manage the system.
- **Database Administration Manual**. This manual should provide the information necessary to enable the data base administrator (DBA) to understand the logical and physical organization, and the record structure of the system.

D3.2.24 Operating Documentation (level 2) Continued:	❑ **Operations Manual**. This manual should provide the operations group with a description of the system operation environment and the detailed instructions they need to execute the system functions.
D3.2.25 Test Reports (level 2):	The project files should be reviewed to ensure that system and acceptance testing results have been documented. Any defects found should have been corrected according to the established procedures that should include the process for assigning, handling, and dispositioning defects.
D3.2.26 Training Plan (level 2):	This document is reviewed to determine if it includes the following elements in the level of detail needed for training staff to begin logistical training arrangements. A "draft" level document is not intended to provide complete training plan information: ❑ Background - system description, objectives, curriculum overview Training requirements - environment, audience, category, skill level of customers ❑ Objective - expected results of the training in terms of the increased level of customer knowledge ❑ Training strategy - type of training, schedule, duration, sites ❑ Training resources - resources required, responsibilities of involved parties ❑ Environment - facilities, support from other groups, equipment, actions required ❑ Training materials - types of materials required, e.g., system reference manual
D3.2.27 Pre-acceptance Checklist (level 2):	Refer to the SOM SDLC document, Section 9, Initiate Acceptance Process, for details and a sample of the checklist.
D3.2.28 Installation Plan (level 2):	This plan is reviewed to assure that it includes the following elements. See SDLC section 8, Installation Plan, for additional details: ❑ General information relevant to the installation process. ❑ Assumptions and dependencies related to the installation activities. ❑ Strategy and schedule for phasing in the new system and disposing of the old one.
D3.2.29 Acceptance Test Plan (final) (level 2):	The acceptance test plan should include the following elements: ❑ The schedule for the test(s) that will be conducted. ❑ A list of the individuals who will conduct the tests and their position/organization. ❑ The test cases and scenarios designed to demonstrate that the requirements have been satisfied in the new system or application.

D3.2.29 Acceptance Test Plan (final) (level 2) Continued:	❑ A list of, or reference to, requirements that the system should satisfy. ❑ Required hardware, software, documentation, special environmental or operational requirements, and any other special considerations (e.g., travel).
D3.2.30 Operational System (level 1):	The reviewer needs to assure that all the system turnover activities have been completed, including the following items: ❑ Execution and completion of the acceptance process. ❑ If deemed appropriate, interviews with the client to determine satisfaction with requirements met. ❑ Final acceptance turnover meeting. This is often waived or cancelled. ❑ Acceptance checklist(s) completed as per the Acceptance process.
D3.2.31 Acceptance Test Report (level 2):	The formal Acceptance Test Report should include a summary of the test procedures executed, any problems detected and corrected, and the projected schedule for correcting any problem reports.
D3.2.32 Maintenance Plan (level 2):	***Note***: This may be part of the Transition Plan. A plan for the support and maintenance of the system after turnover to the customer(s) or after installation into its intended production environment should be documented. It should include the names of the person(s) and/or organization(s) that will provide support, a service level agreement, which process will be followed (presumably the SOM SDLC) for software maintenance, and any schedules, if appropriate.
D3.3 Project Management Review Guidance	Project management includes the set of activities related to planning, organizing, directing, staffing, and controlling available resources to achieve distinct goals and objectives established at the beginning of a project. The key to successful project management is a good project plan, which is developed initially in the planning phase. The plan is dynamic, and should be reviewed and revised in subsequent phases to reflect approved changes.

Section D3.0 Review Guide

Guidance:

This section is a reference guide to assessing the various components of a project plan. The guidance for the following deliverables is provided in this section:

D3.3.1 Project objectives summary
D3.3.2 Development approach
D3.3.3 Project team
D3.3.4 Roles/responsibilities
D3.3.5 Problem escalation
D3.3.6 Assumptions/constraints/dependencies
D3.3.7 Estimates
D3.3.8 Phase/project schedule
D3.3.9 Status reporting
D3.3.10 Resource planning
D3.3.11 Budget - plan vs. actual
D3.3.12 Sign-offs (prior phase exit)
D3.3.13 Configuration management

The table on the following page identifies the appropriate level of review to be performed depending on the project management activity and the phase of development.

Section D3.0 Review Guide

Project Management Review Level Table

		SDLC Phase						
Table Legend	**SDLC Deliverable**	**Planning**	**Requirements Definition**	**Functional Design**	**System Design**	**Programming**	**Software Integration & Test**	**Installation & Acceptance**
Review Levels: ❑ 1 -Verify the existence of the deliverable ❑ 2-Verify minimum content exists ❑ 3-Verify content is rational	Project objectives summary	1						
	Development approach	3						
	Project team	3			3	3	3	
	Roles/responsibilities	2			2	2	2	
	Problem escalation	1						
	Assumptions/constraints/dependencies	3	3	3	3	3	3	
	Estimates	3	3	3	3	3	3	
	Phase/project schedule	3	3	3	3	3	3	
	Status reporting	3	1	1	1	1	1	
	Resource planning	3	1	1	1	1	1	
	Budget – plan vs. actual	2	2	2	2	2	2	
	Sign-offs (prior phase exit)		1	1	1	1	1	
	CONFIGURATION MGT.							
	SCI Identification		2					
	Change initiation		3					
	Change evaluation		3					
	Change approval		3					
	Auditing			2	2	2	2	2
	Change control log			3	3	3		
	Re-baseline requirements			2	2	2		

Section D3.0 Review Guide

D3.3.1 Project Objectives Summary (level 1):

This section defines the project objectives, in summary form. It is intended to give the reader a high level overview of the project. It should be in summary form, and therefore brief (e.g., one-half page to 2-3 pages). It can be copied from other project documents where it may already exist.

D3.3.2 Development Approach (level 3):

This section defines the development environment in terms of the SDLC methodology for the project. This section should include the following elements:

- The methodology that is followed for this project. The standard for SOM Development projects is the SDLC.
- The variations that are taken (if any) from the standard development model.
- If deviations are taken, what is in place to mitigate the increased risk.
- Any feature that is unique regarding the SDLC or project management aspects of this project.

D3.3.3 Project Team (level 3):

The project team section of the project plan is reviewed to assure that the members of a project team have been identified, by name, for the next phase of development and, if possible, for the entire project. The composition of the team should be reviewed and revised as the project progresses through the various phases of development. This section should be reviewed in concert with the section on schedules and estimates to determine if the two sections support each other.

D3.3.4 Roles/Responsibilities (level 2):

The roles and responsibilities of all team members and the customers should be documented initially in the Planning phase. The information should be revised in subsequent lifecycle phases if there are changes, and should be based on the skills and experience of each individual.

This section should include, at a minimum, the names and organizations of those individuals responsible for providing a concur/non-concur position (sign-off) at phase exit. Typically, this would include the project manager's manager, the system owner, the quality assurance representative, and the customer point of contact. Other persons having a support or participatory role should also be identified if possible.

D3.3.5 Problem Escalation (level 1) :

All problems should be brought to the attention of the project manager first. Problem management is the instrument used to handle project-related conflicts, misunderstandings, and problems that cannot be resolved at the project manager level. This section is reviewed to ensure the following items have been identified:

- Escalation point (individual(s) to whom the problem will be brought for resolution if the problem is not resolved at the project manager level.
- Timeframe for a problem to be brought to resolution after it was escalated. This could be one day, several days, or week(s).

Section D3.0 Review Guide

D3.3.6 Assumptions/Constraints/ Dependencies (level 3):

This section should contain all project-wide assumptions, known constraints, and dependencies identified for this project. Assumptions and dependencies should be very specific. Generally known or accepted practices usually do not need to be documented. In many cases, assumptions, constraints, and dependencies will affect project estimates and schedules. This section should be updated and reviewed at every phase of the SDLC.

The following is an example of a dependency:

> The development team is dependent on the availability of customer personnel to review deliverables within ten (10) business days.

The following statement is an example of an assumption:

> Two database administrators with 5 years of Oracle experience will be on board as planned by date.

D3.3.7 Estimates (level 3):

Estimates are reviewed to assure they include the following elements. In subsequent phases, changes to the original estimates may be documented:

- Description of the methodology or combination of methodologies used to arrive at the estimates (planning phase only).
- Line items for all associated project costs including labor months and other direct costs.
- Breakout of project phase activities. This should be detailed for the next phase of development and in summary form for the rest of the project.
- Revisions from the prior phase of development, if there are approved changes that increase or decrease project cost.
- Assumptions upon which the estimates are based.
- Factors used to arrive at the contingency numbers.

D3.3.8 Phase/Project Schedule (level 3):

The project schedule for the next phase is reviewed for evidence of the following attributes. Each activity and its associated data (e.g., start and end dates) can be part of a work breakdown structure that also shows relationships:

- Date for the phase exit.
- Description of each deliverable or activity.
- Individual (name) responsible for the deliverable or activity.
- Projected start and end dates for each activity.

Section D3.0 Review Guide

D3.3.8 Phase/Project Schedule (level 3) Continued:

- ❑ Impact (if any) that approved changes have on the schedule of record (the one approved at the prior phase exit) and the baselining of the new schedule.
- ❑ Critical path(s) identified and dependencies, if any.

For subsequent phases, at a minimum, milestones (phase exit dates) and SDLC deliverables should be documented for each phase.

D3.3.9 Status Reporting (level 3):

Regular, clear, and effective communication is critical to the success of any project. In the planning phase, this section is reviewed to assure the following elements have been identified:

- ❑ Method for reporting project progress and problems
- ❑ Frequency of status meetings
- ❑ Procedure for tracking actions items to closure

In subsequent phases, look for evidence that the established reporting practice is being followed. This might include conference records and meeting minutes.

D3.3.10 Resource Planning (level 3):

Resource planning is essential to the success of a project. It should be started as early as possible in the planning phase, and then revised for each subsequent phase of development. It should be based on the known project requirements and should demonstrate when necessary resources and skills need to be on board.

The following are examples of resources:

- ❑ Hardware: 15 Dell PC's with 40 gigabit of hard drive 128 Mg of RAM and 17" monitors
- ❑ Software: Microsoft Office Suite, Windows 2000 operating system
- ❑ People: two senior analysts, four programmers, two junior programmers, one quality assurance analyst (specify experience requirements)
- ❑ Dates: target dates for acquiring specified resources
- ❑ Office space: Five offices, 12 by 12 feet (each office will accommodate two programmers)

In subsequent phases, look for evidence that action has been taken to ensure resources will be available when needed.

D3.3.11 Budget - Plan Versus Actual (level 2) :

Review this section to verify that it includes the following elements:

- ❑ Project status from the financial perspective.
- ❑ Planned/approved expenditure level to date.
- ❑ Actual expenditure level to date.

The difference (delta) between planned and actual expenditure levels, if any, and to what this difference is attributable. Any deltas should be appropriately reflected in the project plan.

Section D3.0 Review Guide

D3.3.12 Sign-Offs (prior phase exit) (level 1)

The project files should contain the concurrence (sign-offs) received at the prior phase exit. These should be from the individuals identified as having concurrence rights in the roles and responsibilities section of the project plan.

D3.3.13 Configuration Management (level 2):

The configuration management section of the project plan has several important sections that need to be reviewed including the following items:

D3.3.13.1 SCI identification
D3.3.13.2 Change initiation
D3.3.13.3 Change evaluation
D3.3.13.4 Change approval
D3.3.13.5 Auditing
D3.3.13.6 Change control log
D3.3.13.7 Re-baseline requirements

D3.3.13.1 SCI Identification (level 2):

Review this section to verify that software configuration item (SCI) baselines have been identified in order to establish control points for various SDLC deliverables. The following elements are required:

- ❑ Functional baseline (requirements specification)
- ❑ Design baseline (system/subsystem specifications)
- ❑ Production baseline (first version of code)
- ❑ Operations baseline (final version of code)

The training baseline is optional.

D3.3.13.1 SCI Identification (level 2):

Review this section to determine if it contains the following elements:

- ❑ The name and organization of the individual(s) who are authorized to approve requests for change. Ideally, this should be one person such as the customer point-of-contact for the project.

- ❑ The form(s) on which the change request must be submitted. A specific form is not required. Quality Assurance can provide a sample of an appropriate form to use.

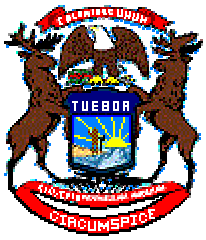
D3.3.13.3 Change Evaluation (level 3):

Review this section to assure that the following elements are identified:

- ❑ The name and organization of the individual(s) who is responsible to evaluate The request for change. In many cases, a change control board is responsible to determine whether a change can be contained within the current project plan.

- ❑ Change evaluation criteria such as the nature of the requested change, time estimates to perform, and an impact analysis on the current schedule and plan.

D3.3.13.4 Change Approval (level 3):	This section should define the procedure for change approval including the following items: ❑ The name and organization of the individual(s) who is authorized to make decisions as to the disposition of evaluated changes. ❑ The acceptable disposition of the change request; e.g., make the requested change, hold the change for a future release, or reject the change.
D3.3.13.5 Auditing (level 2):	This section deals with assuring that the SCM plan is being followed. Look for evidence that demonstrates that change against the previously identified baselines is being managed. Typically, all changes should be recorded in the project's change control log. Other complementary documents such as two-way memos, E-mail notes and messages, and internal forms might also exist.
D3.3.13.6 Change Control Log (level 2):	Review the project's change control log to: ❑ Confirm that change activity is being properly recorded and controlled ❑ Assure that all changes to the project baseline have been evaluated, approved, and noted. ❑ Assure that the impact of accepted changes has been adequately reflected in the project plan, particularly in revised estimates.
D3.3.13.7 Re-baseline Requirements (level 2):	Review this section to ensure that any changes, additions, or deletion of requirements are properly identified and recorded. The new baseline consists of the agreed-to set of requirements from the prior phase, plus or minus any requirements from the current phase of development. The project plan in terms of schedule and budget should reflect the new baseline.



SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX E

PHASE EXIT PROCESS GUIDE

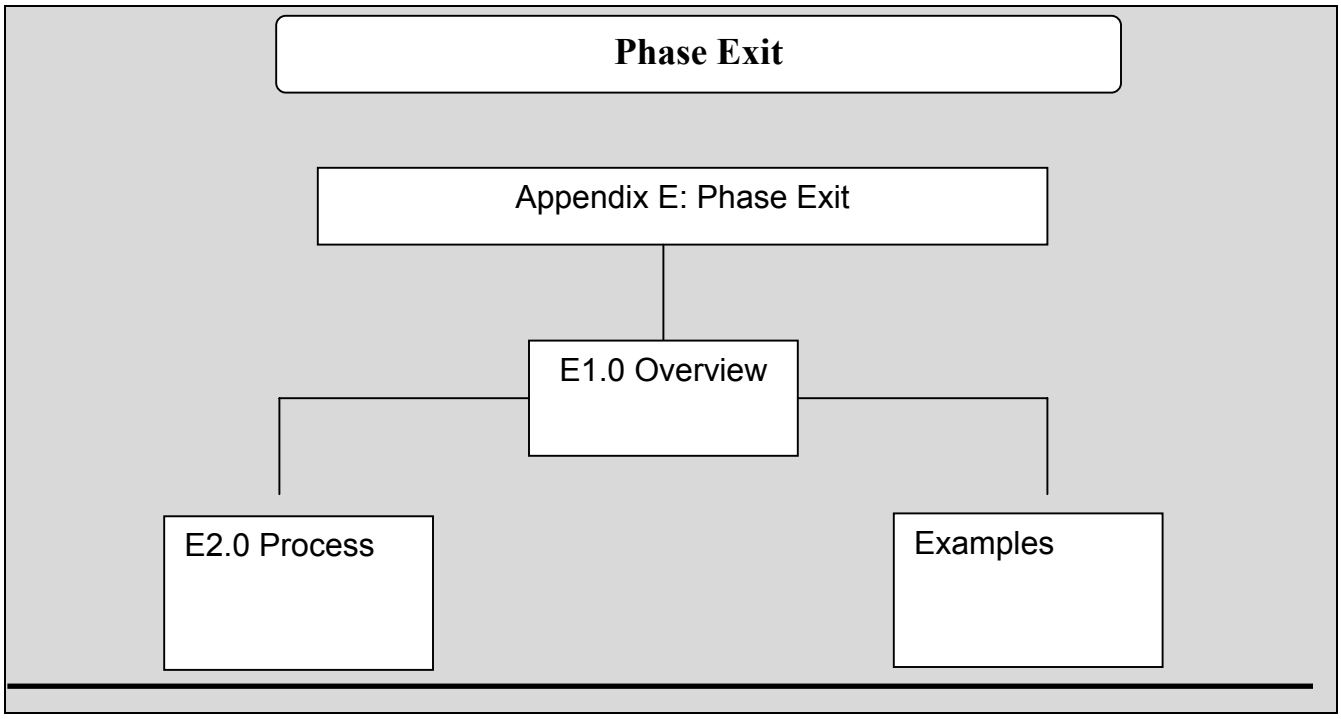
Table of Contents

List of Examples

List of Templates

Section E: Phase Exit

Highlights of Phase



Section E2.0 Process

***Section: E*1.0 Overview**

Introduction:

The SOM Systems Development Lifecycle (SDLC) describes the standard system development lifecycle (SDLC) methodology used for systems developed and maintained for the State of Michigan. For better manageability and control, each system development effort is organized into logical, related segments called phases. Each phase must be officially exited (approved) before the next phase can begin. The decision points at the end of each phase are called Phase Exits.

A Phase Exit is the vehicle for securing the agreement (i.e., approval) of designated individuals to continue with the project and move forward into the next phase of development or maintenance. The agreement is an approval (sign-off) of the deliverables for the current phase of development including the project plan. It indicates that all qualifications (issues and concerns) have been closed or have an acceptable plan for resolution.

***Purpose*:**

The purpose of a Phase Exit is to:

Allow all functional areas involved with the project to review the current project plan. This includes, at a minimum, a detailed plan for the next phase, and high-level plans for the remainder of the project. Provide a forum to raise qualifications (issues and concerns) if issues exist that will impact the project plan.

Ensure an acceptable action plan exists for all qualifications raised. Obtain agreement on current phase deliverables, and to begin the next phase of development.

Applicability:

This process is applicable to all system development and maintenance efforts that follow the SDLC.

Timing/
Frequency:

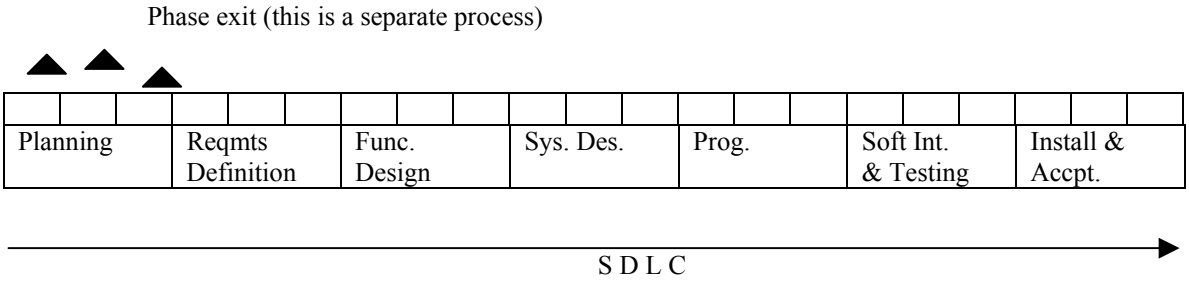
A Phase Exit is conducted at the end of each phase of development or maintenance.

Section E: Phase Exit

Section E2.0 Process

Diagram:

The following figure shows the timing of Phase Exits, relative to the SDLC.



Process Ownership:

The responsibilities of the Phase Exit process sponsor include approving the initial process definition document and changes during process improvement. The responsibilities of the process owner include assuring the process is working once implemented, and conducting ongoing process improvement. The process was originally developed with the support of a cross-functional standards development team.

Change Control:

The Phase Exit process is a component of the SDLC. Changes to this process will be instituted using the same change mechanism that has been implemented to process changes to the SDLC. All requests for change should be directed to the Manager of the site Quality Team.

Relationship to Other System Development Processes:

The Phase Exit process is a primary component of the SOM SDLC. Together with other processes it serves to assure a consistent and predictable outcome in the resulting software products. The Phase Exit process is complementary to other processes such as In-Phase Assessments and Structured Walkthroughs.

Process Measurements:

Process measurements are required in order to understand, in a quantifiable manner, the effectiveness of a process at work. If measurements indicate the process is not working as designed, a causal analysis should be conducted to identify the root cause of the breakdown, and changes should be implemented via the process improvement team. The IPA process is considered to be effective (working as designed) if:

- All issues that must be resolved in the current stage are identified.
- Unmet project objectives can be attributed to issues documented in an IPA.
- All issues without an acceptable action plan become qualifications to exit the current stage of development.
- All issues are properly documented.
- Specific procedures for capturing the data for the above measurements will be defined during process improvement.

Section E2.0 Process

Scope:

The Phase Exit process begins with a notification to the extended development team (e.g., system Owner, customer point-of-contact, support areas) that a phase exit has been scheduled.

The process ends with the receipt of agreement from the designated approvers to proceed to the next phase. Agreement indicates that all known issues have an acceptable plan for resolution.

Customers:

The customers of the Phase Exit process are those individuals or organizations that will use the output of the process. The primary customers are:

- Software development department
- System owner
- Customer point of contact (POC)
- Quality Assurance (QA)
- Information Architecture (IA)
- Agency Security

Suppliers:

The following individuals or organizations provide input to the Phase Exit process:

- System owner
- Project manager's manager
- Customer POC
- QA
- Support areas

Input:

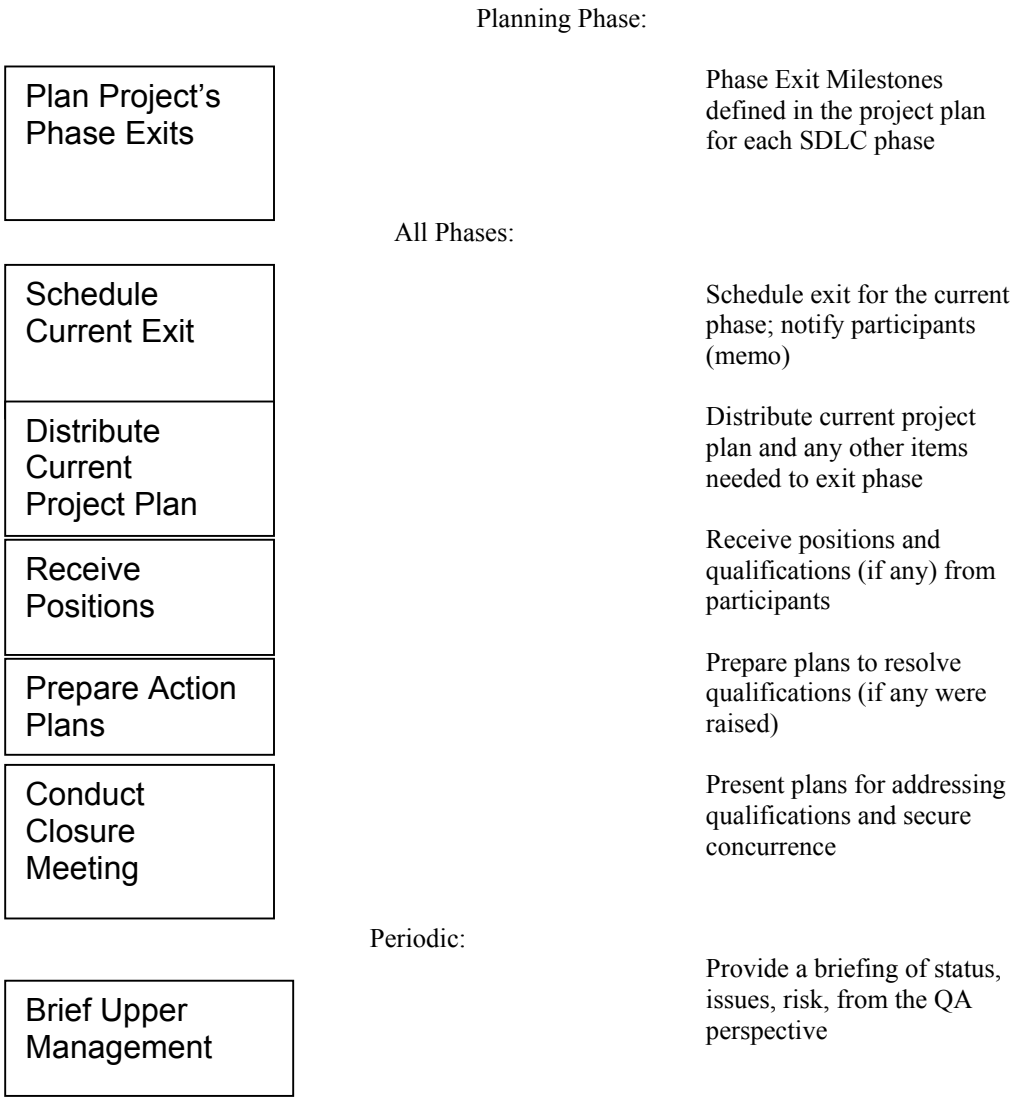
The following are the minimum inputs to the Phase Exit process:

- System development lifecycle deliverable(s)
- Initial project plan developed during the planning phase that includes the work breakdown structure and timeline in addition to other components.
- Updated project plan revised during all subsequent phases
- Structured walkthrough minutes/conference records.

Section E2.0 Process

Diagram:

The following diagram depicts the Phase Exit process flow.



Section E2.0 Process

Plan Phase Exits:
In the Planning phase, the planned date for exiting each phase of development is identified and documented in the project plan. It is common practice for the Phase Exit date for the next phase to be more specific and the dates for subsequent phases to be high level milestones.

Schedule Exit:
For each phase, as soon as practical, the actual Phase Exit date should be established and the exit meeting scheduled. Two or three weeks prior to the exit meeting, a memo is sent to all persons participating in the phase exit to communicate the following information:

- Notify participants that a phase exit has been scheduled. Participants include approvers (e.g., system owner), support area representatives (e.g., Network Engineering), and individuals with a need to know (e.g., contractor management).
- Request that the approvers provide feedback one week before the exit meeting.
- Invite participants to attend the exit meeting.

Examples of a memo, distribution list, and response forms are provided in the example section of this guide.

Distribute Materials:

The current project plan, and any other material relevant to exiting the phase, should be distributed to the participants along with the memo. Relevant materials include for example known issues, and unplanned deliverables.

The participants should be familiar with planned deliverables (e.g. the Requirements document in the Requirements Definition phase) since it is common practice for them to review drafts as they are developed. If this is not the case, then planned deliverables also need to be distributed at this time.

The project plan is dynamic and typically undergoing changes up to the last minute, and is distributed (together or under separate cover) at the same time as the phase exit notification memo.

[1] Project manager is the generic term for the person responsible for planning and day-to-day control of the project, e.g., task leader, team manager, or project leader

[2] In the Planning Phase, participants include all support areas.

Receive Positions:

A position is required from the list of approvers. This position can be agree, agree with qualifications, or disagree. The implication of each is as follows:

- **Agree** - Proceed with the project according to the current plan. An example would be where the approver is not aware of any issues for the current phase.

- **Agree with qualifications** - There are issues or concerns. The project can proceed according to the current plan if an acceptable action plan is developed for each issue by the phase exit meeting. An example would be where there is no plan for testing an interface to an existing system that is being changed.

- **Disagree** - There are very significant issues or concerns. The project should not move to the next phase until issue(s) are resolved. An example would be where funding for the project has been withdrawn or not appropriated.

Section E2.0 Process

All qualifications (issues/concerns) must be communicated to the project manager. The position response form contains space for this purpose, however other forms of communication may be used.

Responses are not required from individuals in the "Support" or "Information" categories of the distribution list; however, they are encouraged to review the deliverables and provide feedback that may have an impact on the project plan.

Prepare Action Plans:

The project manager must prepare an action plan to address each qualification received. Sometimes action plans extend beyond the phase exit milestone. This is acceptable, if it will not negatively impact the current project plan. These action plans are then presented at the phase exit meeting.

Conduct Exit Meeting:

At the exit meeting, the project manager presents positions from the approvers, along with qualifications raised during the phase exit process, and issues that remain open from the In-Phase Assessment (IPA). Action plans must also be presented for each qualification or issue. The objective is to demonstrate that all issues have been resolved, the current plan is sound, and the project is under control.

The results of the meeting are documented in summary form, and include positions, qualifications, action plans, and follow up activity.

Output:

The deliverables developed when the Phase Exit process is executed consists of the following:

- Positions from the approvers
- Qualifications (if any) from review of the deliverables
- Action plans to resolve all qualifications/issues.

Meeting Outcome:

The results of the exit meeting will determine the next step in the development process. The project will proceed in one of the following directions:

- Project proceeds to the next phase according to plan. There were no qualifications raised.

- Project proceeds to the next phase according to plan. All qualifications raised had an acceptable action plan.

- Project cannot proceed to the next phase because significant issues were raised that do not have acceptable action plans to resolve; e.g., funding withdrawn. Schedule a follow-up exit meeting to review action plans and reach concurrence to proceed.

Management Briefings:

Periodically, (e.g. quarterly) the quality assurance analyst will brief the senior or upper level manager (e.g. functional, or contract manager if appropriate) regarding the health and well being of the project, from the QA analyst's perspective. This will minimize the possibility of any surprises later and, if issues exist, they can be addressed timely. The briefing should cover the following areas:

- Project status
- Issue(s) (if any)
- Project risk(s)
- Action(s) required to remove issues or mitigate risk

Section E2.0 Process

Prior to briefing the upper level manager, the QA analyst will brief the project manager and the project manager's manager. This should be more as a matter of courtesy; there should be no surprises, since these persons are involved with the project ongoing.

If serious issues exist, the QA analyst will provide a briefing on an unscheduled basis, rather than wait until the next periodic meeting.

Responsibility Matrix:

The following matrix provides an example of the responsibilities of various parties involved in the Phase Exit process:

	Project Manager (1)	Support Areas (2)	Quality Assurance (QA)	System Owner	Customer (Point of Contact)	Project Manager's Boss (3)	Process Owner/Team
Schedule Phase Exit	P		R				
Distribute current project plan and other materials	P						
Review project plan etc.		R	R	R	R	R	
Agree/Disagree			P	P	P	P	
Prepare action plans	P						
Conduct exit meeting	P		S				
Monitor process effectiveness			P				S
Continuous process improvement	S		S			S	P

P = Perform R = Review S = Support

(1) Project Manager is the generic term for the person responsible for planning and day-to-day control of the project; e.g., task leader, team manager, project leader.
(2) For example, Information Development, Training, Network Planning.
(3) The Project Manager's manager or above.

Example Forms

The following pages provide filled in examples that can be used as a guide for completing the Phase Exit documents.

Example 1

Phase Exit Notification Memo

Date: August 15, 2001

To: Distribution list

From: Jane Doe, Project Manager

Subject: Requirements Definition Phase Exit for the New Inventory System

The Requirements Definition Phase Exit for the New Inventory System has been scheduled. The exit meeting will be held on September 7, 2001 from 9:00 a.m. to 11:00 a.m. at the State Library – Lake Superior room.

Positions, qualifications, and action plans will be reviewed at this meeting.
The review material is attached. A response is required by August 24, from those persons designated as approvers on the distribution list. All others are encouraged to provide feedback and attend the exit meeting. The approver's response may be a position of agree, agree with qualifications (issues), or
disagree. A position form is attached for your convenience. A non-response has the effect of an agreement for that approver.

If you have any questions, please contact me at (517) 241-2926 or Lee Moore at (517) 373-1032 for assistance.

Project Manager
Attachment
cc: Distribution list (attached)
Project notebook/file
(memo only)
As appropriate

State of Michigan
(Insert Agency Name Here)
Example 2 – Phase Exit Distribution List

A. General Information

Information to be provided in this section gives a specific name to the project as well as pertinent information about the personnel involved.

Project Name:		***Date:***	
Controlling Agency:		***Modification Date:***	
Phase:		***Phone:***	
Prepared by:		***Authorized by:***	

B.

Describe the In-Phase Assessment issues and concerns. List how the issues or concern was resolved.

Number	Function	Approval

C.

Describe the In-Phase Assessment correction actions and recommendations.

Number	Function	Support

D.

Describe the In-Phase Assessment correction actions and recommendations.

Number	Function	Information

3
In the Planning Stage, the distribution list must include all support areas. In subsequent stages, only the support areas involved.

Example 3
Phase Exit Participation - LAN Project

	--- Systems Development Lifecycle Phases / Exits ---						
_____ ***APPROVER***	Planning PE	Reqmts PE	Function Design PE	System Design PE	Progmg PE	Soft Intg & Testing PE	Instaln & Accpt PE
System Owner	R	R	R	R	R	R	R
Customer POC	R	R	R	R	R	R	R
Project Mgr.'s Boss	R	R	R	R	R	R	R
Quality Assurance	R	R	R	R	R	R	R
Information Architecture	R	R	R	R	R	R	R
Security	R	R	R	R	R	R	R
SUPPORT AREAS							
Network	N	X	X		X		X
Computer Operations	N						
Database Administration	N						
Systems Development	N						
Documentation	N	X	X	X	X		X
Training	N		X		X	X	X
Records Mgt.	N					X	

PE = Phase Exit.
R = Participation is required.
N = Notification. All areas are notified when the first Phase Exit is scheduled.
X = Sample selection.

Note: This is only an example. The project manager must identify those support areas that need to participate in each phase of development for a given project. The actual list of participants may vary from the example provided.

Example 4
Phase Exit Participation - Mainframe Project

	--- Systems Development Lifecycle Phases / Exits ---						
_____ ***APPROVER***	Planning PE	Reqmts PE	Function Design PE	System Design PE	Progmg PE	Soft Intg & Testing PE	Instaln & Accpt PE
System Owner	R	R	R	R	R	R	R
Customer POC	R	R	R	R	R	R	R
Project Mgr.'s Boss	R	R	R	R	R	R	R
Quality Assurance	R	R	R	R	R	R	R
Information Architecture	R	R	R	R	R	R	R
Security	R	R	R	R	R	R	R
SUPPORT AREAS							
Network	N						
Computer Operations	N			X	X	X	X
Database Administration	N	X			X	X	X
Systems Development	N	X			X	X	X
Documentation	N	X	X	X	X		X
Training	N		X		X	X	X
Records Mgt.	N					X	

PE = Phase Exit.
R = Participation is required.
N = Notification. All areas are notified when the first Phase Exit is scheduled.
X = Sample selection.

Note: This is only an example. The project manager must identify those support areas that need to participate in each phase of development for a given project. The actual list of participants may vary from the example provided.

Example 5
Phase Exit Participation - Client/Server Project

	--- Systems Development Lifecycle Phases / Exits ---						
_____ ***APPROVER***	Planning PE	Reqmts PE	Function Design PE	System Design PE	Progmg PE	Soft Intg & Testing PE	Instaln & Accpt PE
System Owner	R	R	R	R	R	R	R
Customer POC	R	R	R	R	R	R	R
Project Mgr.'s Boss	R	R	R	R	R	R	R
Quality Assurance	R	R	R	R	R	R	R
Information Architecture	R	R	R	R	R	R	R
Security	R	R	R	R	R	R	R
SUPPORT AREAS							
Network	N	X	X		X		X
Computer Operations	N						
Database Administration	N	X	X	X			
Programming	N		X				
Systems Development	N						
Capacity Planning	N		X	X			
Documentation	N	X	X	X	X		X
Training	N		X		X	X	X
Records Mgt.	N					X	

PE = Phase Exit.
R = Participation is required.
N = Notification. All areas are notified when the first Phase Exit is scheduled.
X = Sample selection.

Note: This is only an example. The project manager must identify those support areas that need to participate in each phase of development for a given project. The actual list of participants may vary from the example provided.

Example 6
Phase Exit Position Response Form

******** TO BE SENT TO APPROVERS ********

Project name: New Inventory System (NBS)
Project phase: Analysis
Return form to: Jane Doe, Project Manager, Research and Policy
Return by: August 1, 2002

Position:

- ❑ Agree - Proceed with the project according to the current plan
- ❑ Agree with qualifications - Issue(s) exist. The project can proceed according to the current plan if there is an acceptable action plan for each issue by the phase exit meeting.
- ❑ Disagree - Significant issue(s) exist. The project should not proceed to the next phase until the issue(s) is resolved.

Qualifications (issues):

_____
_____
_____

Approver: _____
Signed: _____ Date: _____

Example 7
Phase Exit Position Response Form

******** RECEIVED FROM AN APPROVER - AGREE WITH QUALIFICATIONS ********
Project name: New Inventory System
Project phase: Analysis
Return form to: Jane Doe, Project Manager, Research and Policy
Return by: August 1, 2002

Position:

- ❑ Agree - Proceed with the project according to the current plan
- ❑ Agree with qualifications - Issue(s) exist. The project can proceed according to the current plan if there is an acceptable action plan for each issue by the phase exit meeting.
- ❑ Disagree - Significant issue(s) exist. The project should not proceed to the next phase until the issue(s) is resolved.

Qualifications (issues):

1. In the requirements document, there are no requirements for expected response times, for both the first and subsequent screens of each transaction.

2. The maximum concurrent number of customers will be 150, rather than 130 as stated in the requirements document.

3. The prerequisite equipment cannot be installed earlier than January 1, 2002. This creates a 2-week variance from the current project plan of record.

Approver: _____
Signed: _____ Date: _____

Templates

The following pages can be copied and used for the Phase Exit documents.

Phase Exit Notification Memo

Date:

To: Distribution list

From: *[project manager]*

Subject: *[phase name]* Phase Exit

The *[phase name]* Phase Exit for *[project name]* has been scheduled. The exit meeting will be held on *[date*], from *[start time]* to *[end time*], at *[location including room number*]. Positions, qualifications, and action plans will be reviewed at this meeting.

The review material is *[attached or has been distributed previously*]. A response is required by *[date]* from those persons designated as approvers on the distribution list. All others are encouraged to provide feedback and attend the exit meeting. The approver's response may be a position of agree, agree with qualifications (issues), or disagree. A position form is attached for your convenience. If no response is received, a position of agree will be assumed for that approver.

If you have any questions, please contact me at *[area code/phone number or electronic address]* or *[alternate name, phone number]* for assistance.
[sign here if hardcopy]

Project Manager

Attachment

cc: Distribution list (attached)
Project notebook/file

(memo only)
[list individuals as appropriate]

Section E2.0 Process

Phase Exit Position Response Form

Project name:
Project phase:
Return form to:
Return by:

Position:

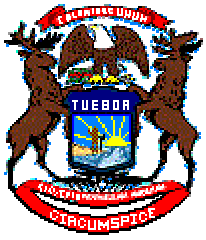
- ❑ Agree - Proceed with the project according to the current plan
- ❑ Agree with qualifications - Issue(s) exist. The project can proceed according to the current plan if there is an acceptable action plan for each issue by the phase exit meeting.
- ❑ Disagree - Significant issue(s) exist. The project should not proceed to the next phase until the issue(s) is resolved.

Qualifications (issues):

_____
_____
_____

Approver: _____

Signed: _____ Date: _____



SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX H

CAPABILITY MATURITY MODEL (CMM)

Section H: Capability Maturity Model

Overview

The Capability Maturity Model (CMM) comes from the Software Engineering Institute (SEI) of Carnegie Mellon University. The SEI has conducted significant research into several areas of business process improvement and reengineering over the past several years.

The CMM, described briefly below, is a professionally recognized model for process development within software-based organizations. The ultimate intent of applying this model for software development in order for State of Michigan agencies to strive to a level five maturity in Project Management. If successful, this practice will eventually spread to other lifecycle areas and throughout Michigan State Government.

CMM Defined

The Capability Maturity Model for Software describes the principles and practices underlying software process maturity and is intended to help software organizations improve the maturity of their software processes in terms of an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes. The CMM is organized into five maturity levels:

1) Initial. The software processes are characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.

2) Repeatable. Basic management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

3) Defined. The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

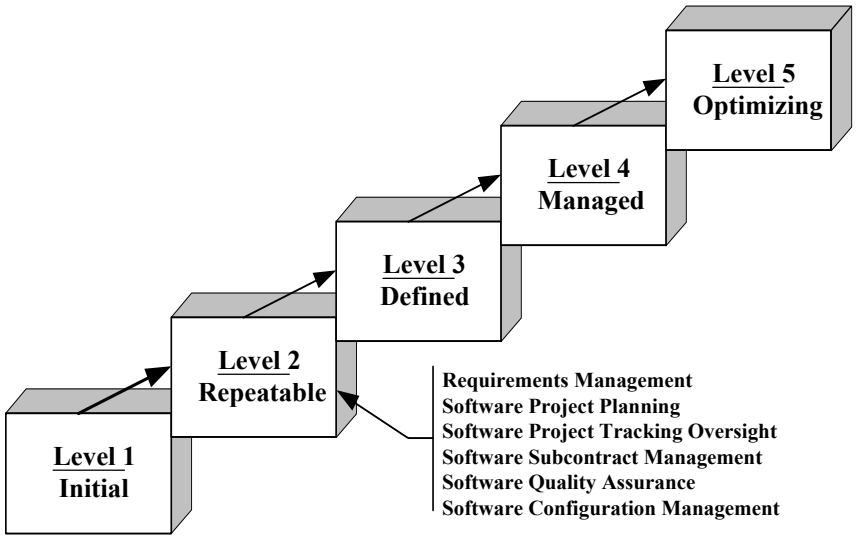
4) Managed. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

5) Optimizing. Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Predictability, effectiveness, and control of an organization's software processes are believed to improve as the organization moves up these five levels. While not rigorous, the empirical evidence to date supports this belief.

Overview

CAPABILITY MATURITY MODEL PROCESS LEVELS



CMM Process Decomposition

Except for Level 1, each maturity level is decomposed into several Key Process Areas (KPA) that indicate the areas an organization should focus on to improve its software process.

The Key Process Areas at Level 2 focus on the software project's concerns related to establishing basic project management controls. They are Requirements Management, Software Project Planning, Software Project Tracking and Oversight, Software Subcontract Management, Software Quality Assurance, and Software Configuration Management.

The Key Process Areas at Level 3 address both project and organizational issues, as the organization establishes an infrastructure that institutionalizes effective software engineering and management processes across all projects. They are Organization Process Focus, Organization Process Definition, Training Program, Integrated Software Management, Software Product Engineering, Intergroup Coordination, and Peer Reviews.

Overview

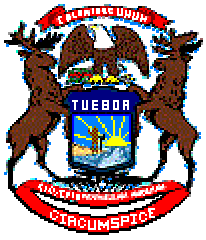
The Key Process Areas at Level 4 focus on establishing a quantitative understanding of both the software process and the software work products being built. They are Quantitative Process Management and Software Quality Management.

The Key Process Areas at Level 5 cover the issues that both the organization and the projects must address to implement continual, measurable software process improvement. They are Defect Prevention, Technology Change Management, and Process Change Management.

Each Key Process Area is described in terms of the key practices that contribute to satisfying its goals. The key practices describe the infrastructure and activities that contribute most to the effective implementation and institutionalization of the key process area.

Finding Additional Information on CMM and the SEI

To find out more about the Capability Maturity Model and implementing its methods into your workplace, you can visit their website at http://www.sei.cmu.edu/



SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX I

COMMERCIAL OFF-THE-SHELF SOFTWARE (COTS)

Section I: Commercial Off-The-Shelf (COTS)

Software Procurement/Implementation Lifecycle

Description:

There is a current trend in systems development to make greater use of Commercial-Off-The-Shelf (COTS) products, that is, to buy a ready-made system from a software manufacturer rather than developing it in-house from scratch. This carries with it a sense of getting a system that can do the job, at a reasonable cost, and getting new function in subsequent releases over time. This practice is especially encouraged, and sometimes mandated, in government agencies. There can be many benefits in using COTS products including improving quality and performance, developing and delivering solutions more quickly, maintaining systems more cost effectively, and standardizing across the organization. The main characteristics of a COTS product are that it exists, is known to be proven, and is available to the general public, and can be bought, leased, or licensed.

COTS and Open Systems:

Many initiatives are under way in both private industry and government agencies, including the SOM, to promote the use of an open systems approach, thereby anticipating even greater benefits than can be obtained from the use of COTS products alone. These initiatives are occurring because just buying COTS does not necessarily result in an "open" system. COTS products are not necessarily open, and they do not necessarily conform to any recognized interface standards. Therefore, it is possible that using a COTS product commits the user to proprietary interfaces and solutions that are not common with any other product, component, or system.

COTS Planning Considerations:

If the sole objective is the ability to capture new technology more cheaply, then the use of COTS products that are not open may satisfy requirements. However, considering that the average COTS component is upgraded every 6 to 12 months and new technology appears on the scene about every 18 to 24 months, any money procuring a COTS product with proprietary interfaces may quickly saves that be lost in maintenance as products and interfaces change.

In the midst of all this, interface standards provide a source of stability. Without such standards every change in the marketplace can impose an unanticipated and unpredictable change to systems that use products found in the marketplace.

A COTS-based systems solution approach requires new and different investments including market research on available and emerging products and technologies, and COTS product evaluation and selection. The key to determining if the best solution is one which includes COTS products is to weigh the risks of straying from the three basic criteria - fully-defined, available to the public, and maintained according to group consensus - against what is to be gained over the long term. An open systems approach requires investments in the following areas early in a project's) lifecycle and on an ongoing basis:

- Market surveys to determine the availability of standards
- Selection of appropriate applicable standards
- Selection of standards-compliant implementations

These costs/activities are the necessary foundation for creating systems that serve current needs and yet can grow and advance as technology advances and the marketplace changes. On an ongoing basis, it is important for project teams to stay informed in this area, with particular focus on:

- When revisions to specific standards are coming
- What changes are proposed in the new revision
- When ballots on the revisions are going to occur
- Where the implementations are headed

Skills Considerations:

The depth of understanding and technical and management skills required on a project team is not necessarily diminished or decreased because of the use of COTS or open systems. Arguably, the skills and understanding needed increase because of the potential complexity of integration issues, the need to seriously consider longer-term system evolution as part of initial development, and the need to make informed decisions about which products and standards are best.

Types of COTS Solutions:

COTS products can be applied to a spectrum of system solutions, including (but not limited to) the following:

- Neatly packaged solutions such as Microsoft Office that require no integration with other components.
- COTS products that support the information management domain, such as Oracle. These systems typically consist of both COTS products and customized components, with some "glue" code to enable them to work cooperatively.
- Systems comprised of a mix of COTS products and non-commercial products that provide large-scale functionality that is otherwise not available. Such systems typically require larger amounts of "glue" code to integrate the various components.

COTS Impact on the Project Lifecycle:

All systems development projects include planning, requirements definition, architecture definition, system design, code, test, and system integration activities. The use of COTS products has an impact on project lifecycle activities. The most fundamental change is that the system is now composed from building blocks that may or may not work cooperatively directly out of the box. The project team will require skilled development expertise to determine how to make a set of components work cooperatively and at what cost.

This fundamental shift from development to composition causes numerous technical, organizational, management, and business changes. Some of these changes are obvious, whereas others are quite subtle.

Requirements Definition:

For a COTS-based system, the specified requirements must be sufficiently flexible to accommodate a variety of available commercial products and their associated fluctuations over time. To write such requirements, the author should be sufficiently familiar with the commercial marketplace to describe functional features for which actual commercial products exist. There is a critical relationship among technology and product selection,

	requirement specification, and architecture definition. If the architecture is defined to fulfill the requirements and then the COTS product is selected, there may be only a few or no available products that fit within the chosen architecture. Pragmatically, three essential elements: requirements, architecture, and product selection must be worked in parallel with constant trade-offs among them.
Adaptation/Integration:	Assembling COTS products presents new challenges. Although software COTS products are attempting to simulate the "plug and play" capability of the hardware world, in reality, they seldom plug into anything easily. Most products require some amount of adaptation and integration to work harmoniously with other commercial or custom components in the system. The typical solution is to adapt each software COTS product through the use of "wrappers," "bridges," or other "glueware". It is important to note that adaptation does not imply modification of the COTS product. Adaptation can be a complex activity that requires technical expertise at the detailed system and specific COTS component levels. Adaptation and integration must take into account the interactions among custom components, COTS products, any non-developmental item components, any legacy code, and the architecture including infrastructure and middleware elements.
Testing:	As the testing of COTS-based systems is considered, it must be determined what levels of testing are possible and needed. A COTS product is a "black box" and therefore changes the nature of testing. A system may use only a partial set of features of a given COTS product. In developing a test strategy and test plans, consideration should be given to issues such as should only the features used in the system be tested, and how does one test for failures in used features that may have abnormal behavior due to unknown dependencies between the used and unused features of a COTS product?
Maintenance:	Maintenance also changes in very fundamental ways; it is no longer solely concerned with fixing existing behavior or incorporating new mission needs. Vendors update their COTS products on their schedules and at differing intervals. Also, a vendor may elect to eliminate, change, add, or combine features for a release. Updates to one COTS product, such as new file formats or naming convention changes, can have unforeseen consequences for other COTS products in the system. To further complicate maintenance, all COTS products will require continual attention to license expirations and changes. All of these events routinely occur. All of these activities may (and typically do) start well before an organization installs the system or a major upgrade. Pragmatically, the distinction between development and maintenance all but disappears.
Adapting the SDLC for COTS Projects:	All systems development projects have a project lifecycle, require project management activities such as project planning, requirements definition, project tracking, configuration management, and quality assurance; and develop deliverables such as project plans, requirements specifications, configuration management plans, and test plans. At the same time, each project, whether COTS or traditional, can vary in scope, duration, technology used, operating platform, etc. The SDLC can be used as the project lifecycle for COTS-based projects as well as for traditional software

Adapting the SDLC for COTS Projects Continued:

development and maintenance projects where all of the code is developed "In-house".

The key to using the SDLC effectively for COTS projects lies in adapting the lifecycle phases and deliverables to best suit the individual needs and characteristics of each particular project. *See Exhibit 2.4-1, Example of SDLC Adapted for COTS Projects*, for an example of how to adapt the SDLC for a COTS project. Phases should be combined as appropriate if, for example, a project will have a relatively small scope, and/or short duration, and/or will use known technology. On the other hand, the traditional number of phases may be appropriate for large projects with new technology and long duration. *Exhibit 2.2-1, Adapting the Lifecycle*, shows how phases can be combined for all types of projects, based on the amount of project management required or anticipated.

Deliverables may be added to, or deleted from the standard list prescribed by the SDLC (*see Exhibit 2.0-1, Software Lifecycle Phases and Deliverables*). For COTS-based projects, the life cycles phases will typically include "Evaluation", "Selection", "Customization", and Integration, and the project deliverables will typically include documents such as "Products to be Evaluated", and "COTS Solution Recommendations".

Documenting Deviations:

The adaptation (or deltas) from the standard SDLC prescribed phases and deliverables are known as deviations. These deviations should be documented with an explanation in the project plan. A statement which describes how project risk is not elevated if a prescribed deliverable will not be developed.

Resources:

The following references are from the features section of the Carnegie Mellon University Software Engineering Institute Web site: http://www.sei.cmu.edu

- Software Technology Review, COTS and Open Systems
- Monthly Features, The Opportunities and Complexities of Applying COTS
- Monthly Features, Discussion with Members of the SEI COTS-Based Systems Initiative
- Software Technology Review, Components-based Software Development/COTS integration

Exhibit 2.4-1. Example of SDLC Adapted for COTS Projects

SDLC Phases	SDLC/COTS Project Phases	Deliverables	
		SDLC/COTS Project Planned Deliverables	Adaptation vs. SDLC Deliverables
Planning	Planning	❑ Project Plan (includes WBS) ❑ Software Quality Assurance Plan	❑ Prototype instead of Feasibility Statement
Requirements Definition	Requirements Definition	❑ Functional Requirements Document ❑ Continuity of Operations Statement ❑ Products to be Evaluated ❑ Software Configuration Mgt. Plan ❑ Data dictionary ❑ Traceability Matrix	❑ The System and Acceptance Test Plans will be developed in the Evaluation and Selection Phase.
Functional Design System Design	Evaluation & Selection	❑ COTS Solution/Recommendation ❑ Acquisition Plan ❑ System Architecture ❑ System/Acceptance Test Plan	❑ Functional Design and System Design phases are combined ❑ System Architecture document replaces Systems Design document ❑ Logical Model, Physical Model, Programming Specifications, Programming Standards not applicable
Programming Software Integration & Testing	Customization, Integration & testing	❑ Solution Baseline ❑ Training Plan ❑ User Documentation ❑ System Maint. Documentation ❑ Transition Plan ❑ Security Plan ❑ System Test Report ❑ System Installation Plan ❑ Pre-acceptance Checklist	❑ The Programming and Software Integration and Testing phases are combined ❑ Integration Test Plan not required
Installation & Acceptance	Installation & Acceptance	❑ Acceptance Test Report ❑ User Training Materials ❑ Acceptance Checklist	❑ No Deviations

Software Procurement/Implementation Lifecycle

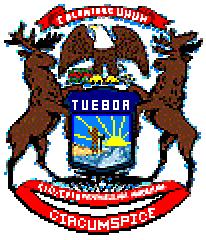
SOFTWARE PROCUREMENT/IMPLEMENTATION LIFE CYCLE				
PHASE	**TASKS**	**DELIVERABLES**	**ROLES**	**TEMPLATES**
Initiation and Planning	Administrative	Ongoing Status Meetings	Project Manager	
	Initiate Project	Project Feasibility Study Project Concept Document Project Charter Core Team Roster Leadership Team Roster	Business Management Project Manager Core Team Leadership Team	
	Write Project Plan	WBS Resource Plan Full Team Roster Schedule Risk Plan Communication Plan QA Plan Procurement Plan Configuration Management Plan Change Management Plan Transition Checklist	Project Manager Core Team Leadership Team	
Requirements and Purchase	Write Objectives and Requirements	Business Process Definition Business Requirements Vendor Analysis Matrix Objectives and Requirements Report	Project Manager Core Team	
	Create Functional Review Test Plan based on business requirements	Functional Test Plan	Project Manager Core Team	
	Perform Initial Software Search	Product Candidate List First Tier Product Review Requirements Finalized	Project Manager Core Team	
	Create Estimate based on: • base product cost • licensing • maintenance fees & release upgrade costs	Purchase Budget Ad Board Approval (if required)	Project Manager Leadership Team	

Software Procurement/Implementation Lifecycle

PHASE	TASKS	DELIVERABLES	ROLES	TEMPLATES
	• training costs • implementation cost • gap analysis (if required) • life cycle costs			
	Create ITB/RFP	Statement of Work Requisition for Software and associated Implementation Services CS-138 (if appropriate) ITB/RFP	Project Manager Buyer	
	Select Software	ITB/RFP Proposal Reviews Second Tier Product Reviews Purchase Order	Project Manager Buyer Contract Administrator Leadership Team Core Team Vendors	
	Analyze Hardware Environment	Preliminary User List Equipment Inventory Equipment Upgrade/ Acquisition Requests	Project Manager Core Team IT Support	
Detail Design/ Gap Analysis	Determine processing and/or software gaps against business requirements Recommend Software Design Changes and/or Process Changes Analyze and Map Data Conversions	Adjusted Schedule Gap Analysis Report Data Conversion Plan Interface Design Report Design/Recommendation Approval	Vendor Project Manager Core Team	
Procedure Development	Draft procedures based on the Gap Analysis and new software	Procedure Manual	Vendor Project Manager Core Team	
Test Plan Development	Identify criteria for testing development, customization, and/or interfaces and data conversions based on draft procedures Develop business-related test	Acceptance Test Plan	Project Manager Core Team	

PHASE	TASKS	DELIVERABLES	ROLES	TEMPLATES
	cases Identify expected results Define test environment issues Create test environment checklist Compile test plan			
Technical Design	Add technical detail to customization plan Add technical detail to interface design Add technical detail to the data conversion plan	Technical Design Report	Vendor	
Technical Development	Develop deliverables listed in any Statements of Work Perform unit and integration testing Delivers executables for Acceptance Test Delivers program documentation	Program Deliverables Technical Test Report Program Documentation	Vendor	
Acceptance Test	Load test data and programsinto test environment Prepare materials for and conduct testing turnover meeting Test all conditions against test system Analyze test results Obtain customer sign-off of program(s) test results	Move Requests Test Results Report Test Approval	Project Manager Core Team	
Training	Develop Training Plan Develop Training Curriculum Develop Course Materials Arrange training site and database use Conduct dress rehearsal, make final revisions, produce materials Conduct training and analyze	Training Plan and Curriculum Course Materials Training Environment Training Evaluations	Vendor Project Manager Core Team	

PHASE	TASKS	DELIVERABLES	ROLES	TEMPLATES
	evaluations			
Implementation	Create Implementation Plan with: • schedule of activities • install/de-install instructions • software dependencies (ex: required ODBC drivers) • configuration information	Implementation Plan	Vendor Project Manager Core Team IT Support	
	Schedule conversion with customers, DBA unit Perform data conversion Verify control reports against converted data	Converted/Loaded Data Conversion Report	Vendor Project Manager Core Team IT Support	
	Build security tables Verify security access	Security Structure	Vendor Project Manager Core Team	
	Software Deployment	Finalized User List Installed Software	Project Manager Core Team IT Support	
	Deliver copy of the final documentation and software to CCM for version control	Configuration Management	Project Manager IT Support	
	Distribute customer codes and passwords to entire customer community	Access Report	Core Team	
Close Out	Provide Life Cycle Maintenance Plan Deliver copy of the final documentation and software to CCM for version control Identify Lessons Learned	Life Cycle Maintenance Plan Configuration Management Lessons Learned Close Out Celebration	Vendor Project Manager Core Team Leadership Team IT Support	



SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX J

DOCUMENTATION STANDARDS

Section J: Documentation Standards

Table of Contents

Section J: Documentation Standards

Highlights of Section

1) ***General Standards***
    1) Purpose
    2) Overview
    3) Organizational Chart
    4) Naming Conventions
        1) Documentation server
        2) Programs
        3) Files
        4) JCL/WFL
        5) Standard Report Headings

2) ***General Procedures***
    1) Purpose
    2) Security
        1) Forms – Hard copy only
        2) Disclosure
        3) Confidential document handling
    3) MDCH MIS Documentation Standards Development Process flow
       MDCH MIS Documentation Standards Development Process narrative
    4) Development Process Initial Project Plan Template
       Development Process Initial Project Plan narrative

3) ***Documentation Standards***
    1) Purpose
    2) MIS-001 Request For Services form
       Request For Services instructions
    3) *Scope*/*Requirements Section*
        1) Contact Person Responsibility memo
        2) *Appraisal Package*
            (1) Development Process Initial Project Plan document (See II.D.)

Highlights of Section

(2) MIS-124 Estimated Project Cost Summary form
Estimated Project Cost Summary instructions
(3) MIS-126 Appraisal of Customer Request form,
which includes the following items:
- Overview item
- Internal/External Entities Affected item
- Contacts item
- Critical Dates with Impacts item
- Recommendation item
- Alternative item

Appraisal of Customer Request narrative

(4) Customer Acceptance of Recommendation Email narrative

(5) System Requirements Package

1 Input and Output Definitions (MIS-020)
Input and Output Definitions narrative
2 MIS-144 Business Rules form
Which includes the following items
- Static data rules
- Data operation rules
- Change event rules
- Authorization rules

Business Rules narrative
3 MIS-146 Business Process Reengineering form
Which includes the following items:
- General description
- “As-Is” scenario
- “To-Be” scenario

Highlights of Section

Business Process Reengineering narrative

4) MIS-050 Customer Acceptance Form
   Customer Acceptance Instructions
5) *Analysis/Design Section*
   1) *Systems Overview Package*
      (1) System Description (MIS-020)
          System Description narrative
      (2) MIS-224 Flowchart form
          Flowchart narrative
   2) *Layout Plan Package*
      (1) MIS-232 Report/Screen Layout form
          Report/Screen Layout instructions
      (2) MIS-234 Output Description form
          Output Description instructions
      (3) MIS 236 Output Content form
          Output Content instructions
      (4) MIS-238 File Description form
          File Description instructions
      (5) 242 Database Design Request form
          Database Design Request instructions
      (6) MIS-244 Request For Database Data Update form
          Request For Database Data Update instructions
   3) MIS-250 Technical Considerations Package form,
      which includes the following items:
          Conversion item
          Customer support item
          Security item
          Technical/Operations support item
          Backup and recovery item
          Training item

Highlights of Section

Implementation coordination item

Library/Storage item

Technical Considerations Package narrative

4) Final/Time Cost Estimates (MIS-124)

Final/Time Cost Estimates instructions

5) Customer Acceptance Form (MIS-050 see D.)

6) *Development/Testing Section*

1) System Testing narrative (MIS-020)

2) Customer Acceptance Form (MIS-050 see D.)

7) *Implementation Section*

1) Implementation Plan narrative/considerations

2) Customer manual narrative

3) *Operations package*

(1) Backup and recovery (MIS-020)

Backup and recovery narrative

(2) MIS-434 Batch Oriented Permfiles (BOP) form

Batch Oriented Permfiles (BOP) instructions

(3) MIS-436BULL Control Block/Microfiche Parameters form

BULL Control Block/Microfiche Parameters instructions

(4) MIS-438 BULL GOS-H Job Stream Description form

BULL GOS-H Job Stream Description instructions

(5) MIS-452 BULL GOS-S Special Retentions/Transfers Off Site Requests form

BULL GOS-S Special Retentions/Transfers instructions

(6) MIS-454 BULL Operations GOS-H Setup Sheet form

BULL Operations GOS-H Setup Sheet instructions

(7) Documentation Checklists narrative

1 MIS-456 BULL documentation checklist form

2 MIS 457 Relational documentation checklist form

3 MIS-458 System documentation checklist form

Highlights of Section

4 MIS-459 Unisys documentation checklist form

(8) MIS-462 File Retention And Disposition form

File Retention And Disposition instructions

(9) MIS-466 JCL/WFL/Program Installation form

JCL/WFL/Program Installation instructions

(10) MIS-468 Key Entry Instructions form

MIS-472 Output Distribution form

(11) Output Distribution instructions

MIS-474 Permanent Disc File Action Request form

(12) Permanent Disc File Action Request instructions

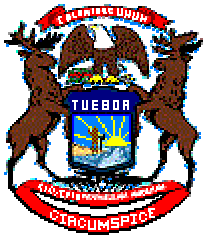
MIS-476Run Instructions form

(13) Run Instructions narrative

Customer Notification Email narrative

8) *Post Implementation Section*
   1) Documentation Storage Requirements narrative
   2) Customer feedback narrative
   3) Request for Services Completion memo example (MDCH memo head)
   4) Project Completion narrative

4) ***Indexes***
   1) Purpose
   2) Forms alphabetical by name
   3) Forms numerical by form number
   4) Forms alphabetical by audience
      1) Systems development staff
      2) Operations staff
      3) Technical support staff
      4) Library staff
      5) FIA staff



SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX L

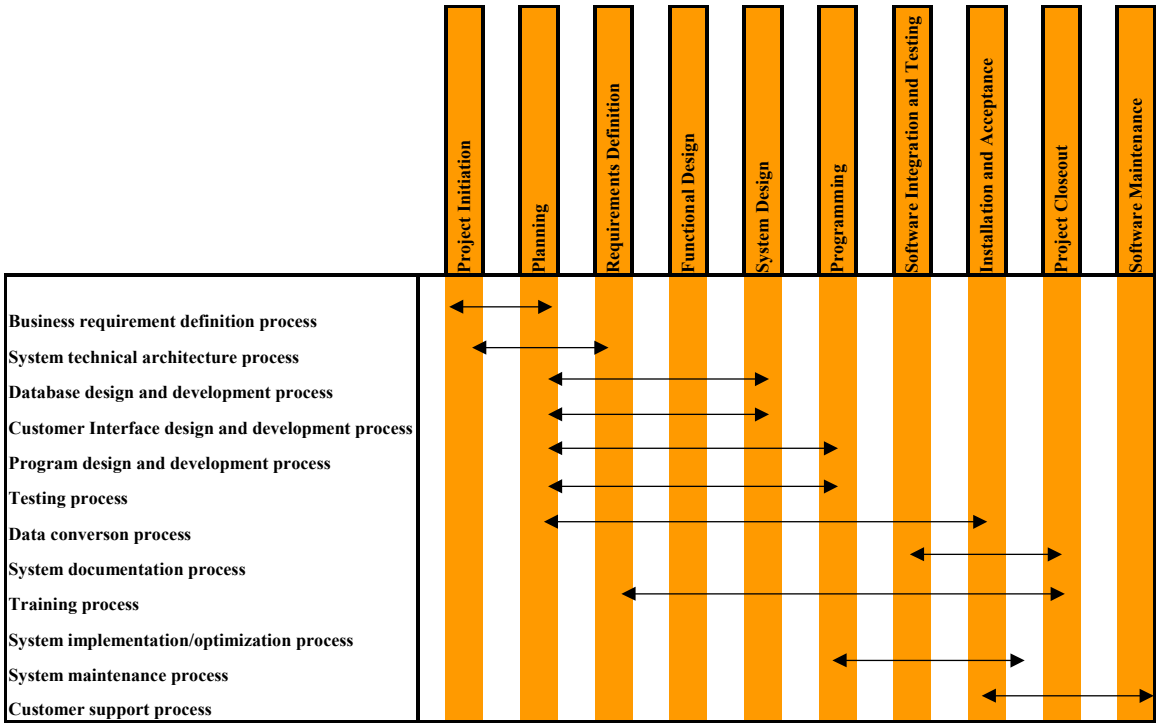
LARGE PROJECTS GUIDE

Table of Contents

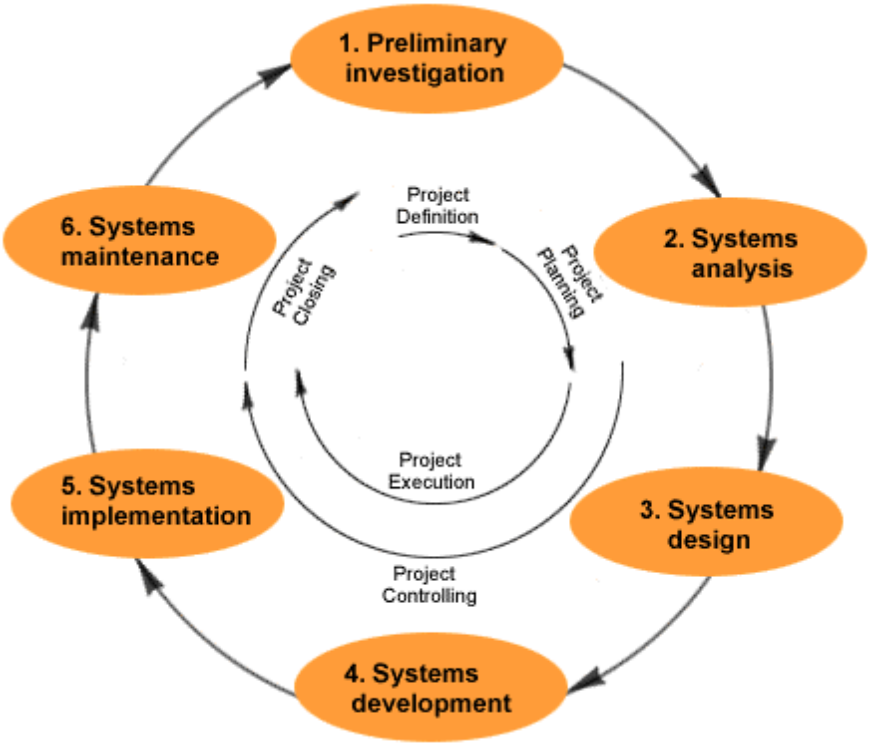
Overview

Description:

Large software development projects are included in the system owner's organizational long-range plans. Enterprise-wide and agency-specific projects are usually developed as large-sized projects and are likely to require a major acquisition of hardware and software. Typically, the larger the size and scope of the project, the greater the detail and coordination needed to manage the project. As risk factors and levels of effort increase, the scope of project management also increases and becomes a critical factor in the success of the project.



Highlights of Section



Highlights of Section

Process	Description
Project Initiation	The purpose of the project initiation phase is to define the customer's business needs, business objectives, project scope, and project constraints and risks.
Planning	
Requirements Definition	The purpose of the analysis phase is to *define the detailed requirements* for the system. • A data and process model is created to capture the *functional and data requirements* of the application. • The *technical architecture requirements* that are required to operate and support the application are also captured at this stage, such at the system security, control, performance, and hardware/software/networking requirements. • A high level *testing, training, and data conversion strategy* is elaborated during this phase. • A working *prototype of the system graphical user interface* is also constructed during this phase and reviewed with the users.
Functional Design	
System Design	The purpose of the design phase is to transform the system requirements into a *set of detailed system specifications*, which must be kept in line with the technical architecture required to run the system. • The data model is transformed into logical database design. • The process model is transformed into detailed and complete set of system transactions (functions) that are carefully designed with their appropriate graphical user interface. • The testing, training, and data

Highlights of Section

	conversion strategies are refined, and detailed plans are produced outlining the activities that must performed to satisfy the identified testing, training and data conversion requirements.
Programming	
Software Integration & Testing	The purpose of the development phase is to *code and test* the system to verify that it works as stated in the system specifications. The test and and production databases are also created. Depending on the number of programs that need to be developed, the system programs can be developed all at the same time or through an incremental series of fast builds.
Installation and Acceptance	The purpose of the implementation phase is to *migrate the system into production environment* and convert the existing data into the new system files and databases. The users who will utilize the system and the Information System personnel who operate the system are trained.
Project Closeout	
Maintenance	System maintenance includes e.g. fixing defects, adding features in updated versions, revisions of the user interface, and responding to development of computing platforms. User support includes e.g. giving instructions and guidance, answering questions, trouble-shooting support. Collection of feedback from users on system error and defects.

Work Breakdown Structure

This work breakdown structure is intended for projects that are 750 hours or greater. This work plan and others are available on the web site at: http://www.michigan.gov/dit

I Project Initiation/Definition Phase

- I-0 Project Initiation/Definition Activity
- I-05 Project initiated
- I-10 Create Project Description Statement
- I-20 Create Project Concept Document
- I-30 Project Concept Document Completed
- I-40 Create Project Appraisal Package
- I-50 Project Appraisal Package Approved
- I-60 Create Project Feasibility Document
- I-70 Project Feasibility Document Completed
- I-80 Create Project Charter Document
- I-90 Project Charter Document Approved
- I-100 Steering Committee Approval
- I-120 Project Management
- I-130 Manage Project
- I-140 Initiation Completed

P Planning

- P-0 Objectives and Scope
- P-1 Planning Started
- P-2 Develop Critical Success Factors
- P-3 Develop Project Scope Statement
- P-4 Project Scope Statement Developed
- P-5 Develop Project Work Plan
- P-6 "Develop WBS, Assign Resources, Schedule"
- P-7 Develop Organizational Breakdown Structure
- P-8 Develop Cost Benefit Analyses
- P-9 Project Work Plan Developed
- P-10 Develop Procedural Plans
- P-11 Develop Configuration Management Plan
- P-12 Develop Risk Plan
- P-13 Develop Quality Plan
- P-14 Develop Communications Plan
- P-15 Procedural Plans Developed
- P-16 Requirements
- P-17 Prepare Input/Output Definitions
- P-18 Document Business Rules
- P-19 Develop Business Reengineering Proposal
- P-20 Customer Accepted Requirements

P-21 Analysis/Design
P-22 Prepare Systems Overview Documents
P-22-1 Prepare Web Design
P-23 Prepare Database Design
P-24 Prepare Screen Layout
P-25 Prepare Report Layout
P-26 Prepare Program Overview
P-27 Prepare Technical Considerations Document
P-28 Develop Final Time/Cost Estimates
P-29 Revise Project Plan and Rebaseline
P-30 Customer Accepted Design
P-31 Implementation Planning
P-32 Develop Initial Implementation Plan
P-33 Develop Initial Training Plan
P-34 Complete Initial Transition Checklist
P-35 Initial Implementation Plan Approved

E Execution

E-0 Program Instructions
E-1 Write Programming Instructions
E-2 Programming Instructions Completed
E-3 Development
E-4 Create Database
E-5 Write Programs
E-6 Create Screens
E-7 Create Reports
E-8 Conduct Unit Testing
E-9 Development Completed
E-30 System Testing
E-32 Test System
E-35 Testing Completed
E-45 Customer Acceptance Testing
E-50 Prepare User Manual
E-55 Prepare Test Environment
E-60 Assist Customer with Acceptance Testing
E-65 Code/test Corrections
E-70 Assist Customer with Regression Testing
E-75 Review Implementation Plan
E-80 Review Training Plan
E-85 Review Transition Checklist
E-89 Customer Accepted System
E-90 Implementation
E-91 Execute Implementation Plan

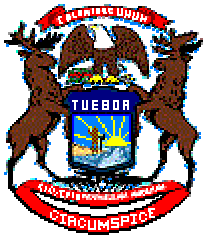
E-95 Execute Training Plan
E-99 Implementation Completed

C Control

C-0 Project Control Activities
C-1 Project Control Started
C-2 Update Project Work Plan
C-3 Conduct Status Meetings
C-4 Perform Metrics and Status Reporting
C-5 Perform Change Control
C-6 Perform Scope Control
C-7 Perform Quality Control
C-8 Perform Schedule Control
C-9 Perform Cost Control
C-10 Perform Risk Control
C-11 Perform Contract Administration
C-12 Contingency Budget
C-13 Project Control Completed

CL Closeout

CL-0 Administrative Closure
CL-1 Perform Administrative Closure
CL-2 Perform Financial Closure
CL-3 Perform Financial Audit
CL-4 Archiving
CL-5 Personnel and Facilities
CL-6 Project Administratively Closed
CL-7 Post Implementation
CL-8 Post Implementation Support
CL-9 Document Lessons Learned
CL-10 Assemble Post Implementation Evaluation
CL-11 Project Completed



SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX M

MEDIUM PROJECTS GUIDE

Section M: Medium Projects

Table of Contents

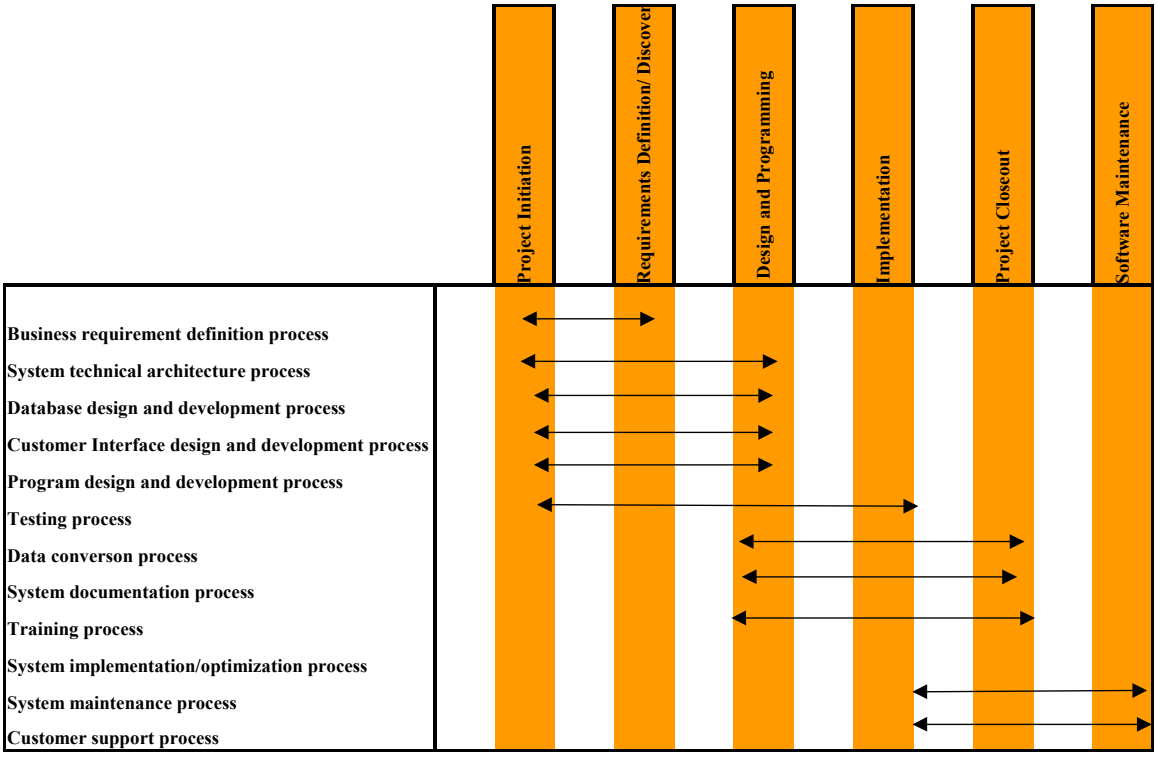
Section M: Medium Projects

Highlights of Section

Description:

Medium software development projects require less effort than large projects, typically use existing hardware and software, and might not be captured during the organizational long-range planning process. Medium size projects are frequently developed to automate operations within a programmatic office or among a limited number of sites, and may be used to interface with other software products.

Planning medium size projects within the context of the system owner organization's overall mission, and building in compatibility to the Agency computing environment can improve the software product's ability to interface with other customers, organizations, and applications; and increase the product's longevity.

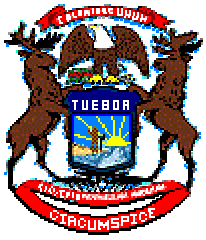


Highlights of Section

Process	Description
Project Initiation	The purpose of the project initiation phase is to define the customer's business needs, business objectives, project scope, and project constraints and risks.
Planning	
Requirements Definition	The purpose of the analysis phase is to *define the detailed requirements* for the system. • A data and process model is created to capture the *functional and data requirements* of the application. • The *technical architecture requirements* that are required to operate and support the application are also captured at this stage, such at the system security, control, performance, and hardware/software/networking requirements. • A high level *testing, training, and data conversion strategy* is elaborated during this phase. • A working *prototype of the system graphical user interface* is also constructed during this phase and reviewed with the users.
Functional Design	
System Design	The purpose of the design phase is to transform the system requirements into a *set of detailed system specifications*, which must be kept in line with the technical architecture required to run the system. • The data model is transformed into logical database design. • The process model is transformed into detailed and complete set of system transactions (functions) that are carefully designed with their appropriate graphical user interface. • The testing, training, and data

Highlights of Section

	conversion strategies are refined, and detailed plans are produced outlining the activities that must performed to satisfy the identified testing, training and data conversion requirements.
Programming	
Software Integration & Testing	The purpose of the development phase is to *code and test* the system to verify that it works as stated in the system specifications. The test and and production databases are also created. Depending on the number of programs that need to be developed, the system programs can be developed all at the same time or through an incremental series of fast builds.
Installation and Acceptance	The purpose of the implementation phase is to *migrate the system into production environment* and convert the existing data into the new system files and databases. The users who will utilize the system and the Information System personnel who operate the system are trained.
Project Closeout	
Maintenance	System maintenance includes e.g. fixing defects, adding features in updated versions, revisions of the user interface, and responding to development of computing platforms. User support includes e.g. giving instructions and guidance, answering questions, trouble-shooting support. Collection of feedback from users on system error and defects.



SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX N

SMALL PROJECTS GUIDE

Section N: Small Projects

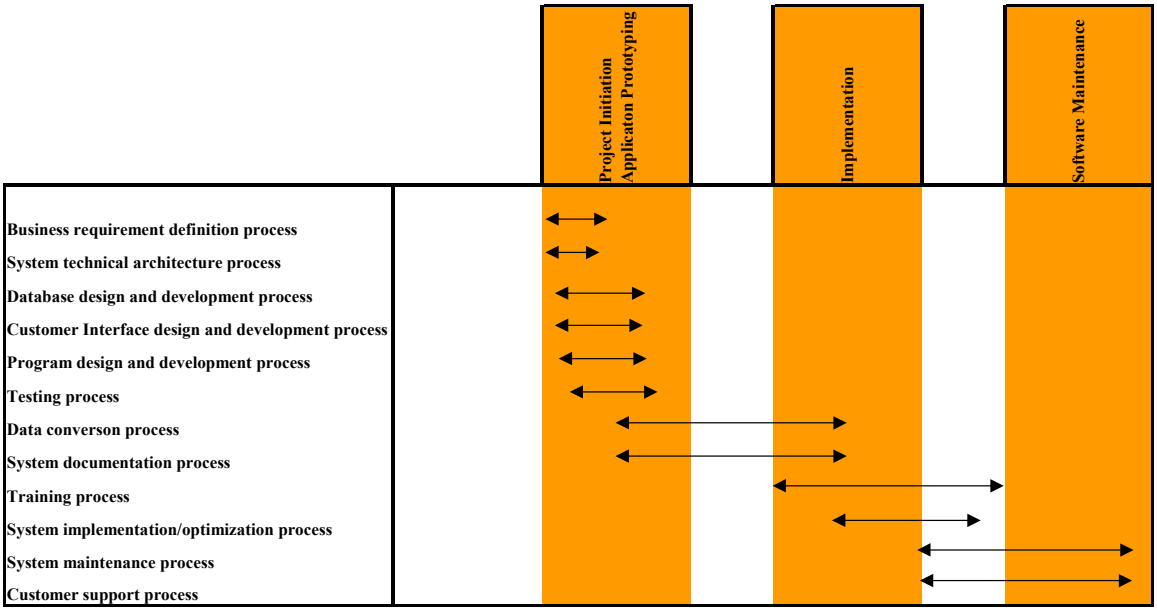
Overview

Section N: Small Projects

Overview

Description:

Small software development projects require minimal effort and use existing hardware and software. The project manager can easily manage the operational details of a small project, so formal documentation requirements are limited. A project is small when the software being developed will have limited functionality and use, meets a one-time requirement, or is developed using reusable code.



Overview

Process	Description
Project Initiation	The purpose of the project initiation phase is to define the customer's business needs, business objectives, project scope, and project constraints and risks.
Planning	The purpose of the planning phase is to
Requirements Definition	The purpose of the requirements definition phase is to *define the detailed requirements* for the system. • A data and process model is created to capture the *functional and data requirements* of the application. • The *technical architecture requirements* that are required to operate and support the application are also captured at this stage, such at the system security, control, performance, and hardware/software/networking requirements. • A high level *testing, training, and data conversion strategy* is elaborated during this phase. • A working *prototype of the system graphical customer interface* is also constructed during this phase and reviewed with the customers.
Functional Design	The purpose of the functional design phase is to
System Design	The purpose of the system design phase is to transform the system requirements into a *set of detailed system specifications*, which must be kept in line with the technical architecture required to run the system. • The data model is transformed into logical database design. • The process model is transformed into detailed and complete set of system transactions (functions) that are carefully

	designed with their appropriate graphical customer interface. • The testing, training, and data conversion strategies are refined, and detailed plans are produced outlining the activities that must performed to satisfy the identified testing, training and data conversion requirements.
Programming	The purpose of the programming phase is to
Software Integration & Testing	The purpose of the software integration and testing phase is to *code and test* the system to verify that it works as stated in the system specifications. The test and and production databases are also created. Depending on the number of programs that need to be developed, the system programs can be developed all at the same time or through an incremental series of fast builds.
Installation and Acceptance	The purpose of the installation and acceptance phase is to *migrate the system into production environment* and convert the existing data into the new system files and databases. The customers who will utilize the system and the Information System personnel who operate the system are trained.
Project Closeout	The purpose of the project closeout phase is to
Maintenance	System maintenance includes e.g. fixing defects, adding features in updated versions, revisions of the customer interface, and responding to development of computing platforms. Customer support includes e.g. giving instructions and guidance, answering questions, trouble-shooting support. Collection of feedback from customers on system error and defects.

Small Project Method

Description:

Small Project Method

The small project method condenses the larger departmental SDLC method into five phases and a management layer. These phases represent a standard configuration of the tasks and documentation for projects not subject to the SDLC threshold criteria; the phases below form the base from which the project team may modify tasks and documentation as required in order to address the needs of their particular project.

Project Management:

The project management includes activities that have been considerably simplified from the activities in the large development approaches.

General Management tasks include:

Define, document, and get appropriate sign-offs on project scope.

Manage personnel resources:

- Set up project advisory group
- Manage day-to-day activities
- Manage risk actively
- Monitor key progress indicators

Maintain customer relationship:

- Establish sign-off process
- Maintain communications with project advisory committee
- Manage expectations

Manage issues, open points and change requests

- Identify and log items
- Assign responsibility
- Facilitate and monitor resolution process
- Report item status
- Escalate items to sponsor or advisory committee
- Communicate resolutions

Requirements Definition:

The small project method requires definition activities in order to understand the requirements, including their structure and organization. This method combines activities from requirements gathering and analysis, reflecting the conditions of a small team of experienced personnel working with a single business process.

Analysis includes several high-level activities including the identification of customer requirements and development of events, data, and process prototype.

Requirements Definition Continued:

Customer requirements analysis

This analysis is intended to understand what the new application must do to satisfy the needs perceived by its customers.

General customer requirements analysis tasks include:

- Planning and conducting interviews (may be either one-on-one interviews or Joint Application Development (JAD). For further information on JAD see the "References" in the VI REQUIREMENTS SPECIFICATION AND PROTOTYPE phase of the DEHNR SDLC. Determine what information to collect. Select interviewees. Schedule sessions.

- Where appropriate conducting external reviews. May include review best practices, site visits of comparable organizations, review of packaged software for additional functionality.

- Performing gap analysis by:

Documenting present business process
Designing/reviewing preferred business process
Highlighting system changes required to bridge the gap resulting from new workflow and organization, resulting from present operational difficulties, and resulting from present technical deficiencies.

Events analysis (optional)

This analysis is intended to model the external behavior of the application according to its business transactions and events.

General events analysis tasks include:

- Based on the scope of the project identifying a list of external events including transactions, alerts such as inventory levels, and sensor events such as time.

- For each event, determining stimulus, describing overall response and any generated events. Iterating as required. Annotating event-stimulus-response description with volumes, frequencies, and desired response times.

- For each major entity, creating an entity life cycle diagram (See data analysis below.) If required, creating an entity by event matrix. Verifying completeness of event model relative to the requirements, relative to present system, relative to data and process models.
- Validating event model with customers.

Data analysis

This analysis is intended to model the application's data requirements, independent of implementation details and to provide a basis for designing the application's database and file structures.

Requirements Definition Continued:

General data analysis tasks include:

- Defining the scope of the data model.
- Defining entity types, type hierarchies, and a unique identifier for each entity type.
- Defining relationship types based on business rules and policies.
- Defining attribute types.
- Normalizing and verifying the completeness of the data model relative to the customer requirements, present system, and event and process models.
- "Scrubbing" existing data.

Document and confirm requirements

Gather and model the requirements into a requirements specification. This specification captures the events, data, and process aspects of the system, as well as its quality attributes, all of which are key inputs to the design process.

Business Process Prototype

This analysis is intended to create the basis for customer approval of requirements on *shown features* rather than written descriptions, to confirm that the specified level of usability can be achieved, and to ensure that innovative or high-risk features are functionally and technically feasible.

General prototyping tasks include:

- Defining scope of prototype
- Developing the prototype review plan
- Selecting the prototyping environment
- Designing the prototype. Pay particular attention to the overall decisions that affect the customer interface metaphor, levels of customer expertise to support, support and control of the work flow, hardware, and software platform (in particular hardware features such as large screens, scanners and other special-purpose devices), and implications for the execution architecture
- Selecting events, processes and data to be prototyped
- Building the prototype
- Applying the methodology prescribed by the prototyping tool
- Testing and reviewing prototype with selected customers. Documenting reactions and requested revisions, and iterating as required.

Since the main goal of a prototype is to gauge the customers' reaction to the future application, develop several sections of the application to provide data entry and retrieval capability, full navigation, etc.

Small Project Method

Detailed Design:

The Design phase of the small project method uses the requirements specification as the basis for system design. Design activities in the small project method are heavily influenced by the use of existing development and execution architectures. These activities reduce technical risk to a certain degree, reducing the need for architecture and several other technical design tasks. The activities in the design phase produce detailed designs of the customer interface, database and processing for implementation in the Build Phase.

General detailed design tasks include:

- Designing dialogue features including the customer interface and reports and documents.

- Designing the application architecture including the messaging and processing flow and the logical and physical database. The purpose of this task is to identify all programs in the application, their sequence, and how they communicate.

Typical steps followed in this task include:

- Identifying client, server, traditional online, asynchronous and batch Programs

- Identifying processing sequence and inter-program communication

- Identifying interfaces to other systems

- Designing program-to-program, internal program, and file-generation controls

- Identifying additional programs required for ancillary processes and iterations.

Designing logical and physical databases. Typical steps followed in this phase include:

Review of existing databases and distribution approach.

For each entity type, choose the data management software (for example,
relational DBMS, flat file)

Identify tables and for each table: 1) Define all columns. 2) Define primary key, 3) Evaluate row length and adjust as needed.

Design foreign keys and referential integrity rules

Design logical database file structures.

Detailed Design Continued:

Review design with appropriate management and staff

Design physical database: relational database management system
(RDBMS) structures.

- Designing the approach to testing ensures that all the time and resources required for testing will be available during the test phase of the small project path. Typical steps in this task include:

Identifying the required testing phases (from unit test through acceptance test)

For each testing phase, determining:

Objectives

Requirements against which to test

Who will execute?

Environment

Approach to regression testing (testing to assure changes in one unit of the
system does not affect other units.)

Approach to creating, maintaining and sharing test data

Tool requirements

- Confirm tool availability

Building System:

This phase produces tested executable code, procedures, training and the system test components. The project team writes and checks the code, then prepares test data and expected results that will fully explore all of the operational conditions that an individual module will face. The team then runs the unit and phase tests, checking the results against those that were expected, and corrects any errors. In parallel with these activities, the team creates the customer procedures and training, and prepares to begin the Test phase.

General Building tasks include:

- Developing and documenting systems performance, security, control and operations procedures with an identification of anticipated customers, drafting of documentation, and development of help materials.

Building System Continued:

- Generating and coding work units. To produce executable code for each designed work unit, review the specifications and complete the programming work units. Desk-check the code, developing any unit-testing routines, compiling until correct, and updating the development status.

- Conducting unit testing to ensure that the programming work units properly perform all required functions. Execute each test cycle. Check results. Verify actual to expected results. Make changes, and retest as necessary.

- Performing code reviews. (optional) To identify where the program does not conform to specifications and to ensure that all code is written and documented according to project standards. Assign reviewers and arrange meetings, document the results, clear the points, notify reviewers of changes, and update the development status.

- Preparing system tests. To ensure that the system test will be thorough finalize dates and assign people, confirm requirements such as hardware time requirements, prepare the test data, finalize the test conditions, and define the test cycles. System testing for small projects refers to the integration and customer acceptance tests.

Testing:

In this phase, the project team is responsible for testing the new system and training the personnel in preparation for production. Test the configurations promoted from programming and review the results. Then conduct an integration test and monitor the system's performance, making changes as necessary. Investigate and correct discrepancies in the results. The objective in this phase is to ensure customers and operations personnel know how to use and run the system and to verify before conversion that the new system contains all required functions that work correctly.

General Testing tasks include:

- To verify that the programs in the new system communicate properly with each other review the test plans (These plans should be developed with substantial customer involvement including preparation of the test data.), verify that resources are available, develop and execute test cycles, verify results, and obtain sign-off.

- Selecting/promoting the system configuration (e.g. selecting and organizing the components of a system.) To promote the system configuration define the configuration, confirm the availability of the system components, verify that the target environment is ready, and select the method to be used to promote the system (i. e. move the system from the test environment to the production environment.) Promote the configuration and verify that the promotion has been successful.

- To verify that the new system can operate effectively in a full production environment perform the system quality and integration tests. Follow the same general test steps listed above.

Testing Continued:

- To ensure customers and operations personnel know how to use and run the system confirm the training session agenda, conduct the training sessions, evaluate the training sessions, and establish a continuing training program.

- To ensure that the system is ready to be rolled out: Conduct a final quality assurance review that reviews the tests and the implementation plans.

Implementing Systems:

The small project method assumes the implementation occurs for an initial site, with subsequent efforts used to execute and manage implementations to multiple sites. A successful implementation involves more than just using the new programs and procedures. The work environment must reflect only the new system; remnants of the old system must be removed when it makes sense. The project team must review and analyze the system to verify that it performs the functions stated in the design. Finally, the project team must document, for future implementation, any changes or enhancements identified during the initial production cycles that would make the new system more responsive to customer and business needs. The small project method defines the implementation process for an individual site. Repeat the activities in this phase for each distributed site and once for a central site.

General Implementation tasks include:

- To verify that operations personnel can effectively support the new system and to ensure customer acceptance of the new system, perform a readiness test. The general steps are the same as those used in earlier testing tasks.

- To convert the site, develop conversion plan, verify that prerequisites and resources are in place, review conversion process with conversion team, resolve open issues, convert files and “scrubb” data where necessary, verify results, remove old system, establish customer liaison, develop a maintenance manual, and transfer responsibility.

- To ensure that the new system is functioning as designed provide support during initial production, monitor progress, record and control variances, make adjustments where necessary. Document all change requests.

Rapid Application Development:

Rapid application development (RAD) integrates project management techniques, development techniques, customers and tools to build quality application systems in a fixed time frame to deliver business value. Rapid development combines very focused teams working in a highly structured environment. A rapid development team consists of between three and seven people focused on creating a specific application from a prioritized and fixed set of requirements. Certain types of applications lend themselves to this development approach. See RAD criteria to determine project applicability.

Rapid Application Development Continued:

Rapid development is a focused process in which the conceptual requirements of the application are fed into construction iterations. The construction iterations occur within a fixed timebox that regulates the resources that can be expended during the iterations. The iterations ultimately result in the system that is rolled out to the production sites.

Rapid development consists of three phases and a management layer which support the application development process from project planning through implementation. An iterative development approach is applied to construction activities with extensive use of evolutionary prototyping. Joint Application Development (JAD) is used to collect and prioritize application requirements. (See Reference VI.D.3 for an expanded discussion about JAD.)

The Manage layer in rapid development is responsible for project organization, project management and scope control. This layer also covers the management of the project's business case. Activities in the Manage layer are used to oversee rapid development's core process activities, including establishing the project team, the scope and organization of the construction process, and the conversion and roll out plans. Techniques related to timeboxing and construction iterations are found in this phase.

General steps in the manage project and business case task inlcude:

- Organize RAD project, confirming the project's overall scope and objectives, establishing executive sponsorship, establishing the initial business case, identifying and classifying appropriate customer characteristics and types, identifying appropriate customer and IS personnel for the rapid development team, and obtaining commitments for their involvement, establishing a work plan for rapid development activities, and orienting the rapid development team and initiating the project.

- Manage project analysis.

- Establish “Timebox” iterations by determining the overall scope of the project based on the requirements, selecting an organizational approach for the construction iterations, determining the scope and objective of each iteration, defining the testing approach for each iteration, confirming the overall approach and scope with the sponsor and project team, and determining the migration approach.

- Manage project construction by identifying and resolving issues, communicating status to executive sponsor, coordinating project team use of IS resources and personnel, and assessing the progress of the project team against the work program.

- Plan and manage the project implementation by organizing implementation and training
activities, identifying and resolving issues, communicating status to executive sponsors, and coordinating project team use of IS resources and

Analyze:

personnel

Identifying the customers business needs, the related requirements and the priorities of those requirements is the primary objective of the Analyze phase. This phase includes tasks and techniques related to identifying and prioritizing system requirements that define the scope and functionality of the system. Joint Application Development (JAD) supports the requirements definition activities in this phase.

General steps to conduct the analysis task include:

- Creating an Event Model (See the "Events Analysis" section" of the SMALL PROJECT SDLC.)

- Identifying Customer Requirements (See the "Customer Requirements Analysis" section of the SMALL PROJECT SDLC.)

- Prototyping Business Process (See the "Business Process Prototyping" section of the SMALL PROJECT SDLC.)

- Creating a Data Model Process (See the "Data Analysis" section of the SMALL PROJECT SDLC.)

- Creating a Process Model (i.e. to complete the internal behavior of the application in response to business transactions and events) by selecting the process modeling approach (i.e. Data Flow Diagramming, Business Function Decomposition and Elementary Process Description), developing the process model, verifying the completeness of the process model (e.g. relative to the requirements, relative to the data and event models) and validating the process model with customers. The process model forms the basis for the design of application code.

Rapid development uses an iterative approach to the design, build and test activities in the Construct phase. These iterations, managed through a timebox, give the project team the flexibility needed to implement the system in increments rather than in a single effort. The construction process supports the implementation of the requirements according to their value or risk and incorporating customer comments and requirements into the system.

Construct:

The general construct tasks include:

- **Customer Interface design:** The customer interface should support the business processes. It incorporates the design of the components visible to the customer: systems windows, reports and forms. In rapid development, the customer interface design should use feedback generated from the business process prototype.

- **Messaging and processing flow design**: In this phase, one designs internal program, program-to-program, and file-generation and maintenance controls, identifies additional programs required for ancillary processes, and

Construct Continued:

documents resulting processing flow. This is an iterative process.

- **Database design:** The rapid development teams use of existing database management systems is assumed as one of the entry criteria for this development approach. Existing technical architecture components like the DBMS and existing production databases, in addition to the highly focused project scope manages the breadth of the database design. These factors support condensing the logical and physical data base design into this single task.

- **Data and process network distribution design:** Data and process distribution are primary concerns associated with developing client/server and other forms of distributed systems. Of course, you would omit this task if you are developing systems whose data and process will be centrally located, such as host based systems.

- **Generate and code work units:** To execute this step review the specifications, generate program components including completion of the programming work units, documentation of potential test conditions, conduct unit-testing routines, compile until correct, and perform design or peer and customer reviews. Iterate as necessary. Rapid developments use of existing technical architectures and development environments are intended to reduce the complexity and risk associated with coding and generation. The rapid development team should seek to leverage the capabilities of the application development environment and the application architecture in order to focus their efforts on correctly implementing the requirements.

Implementation:

The Roll Out phase follows the completion of the construction effort. It includes creating procedures, training the customers and converting the application into production.

The general implementation tasks include:

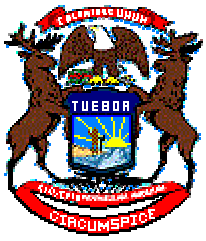
- Identification of procedures and development of operations manuals.

- Development of training materials, train instructors, and conduct training sessions as needed.

- To verify that programs in the new system communicate properly with one another, prepare testing plans, execute test cycles, verify results, make changes and retest as necessary, and obtain sign-off.

- To transfer the system to operational status, prepare conversion plan verifying prerequisites and resources are in place, reviewing plan with conversion team, resolving open issues, converting files, verify results, and set up transfer support group to remove the old system and transfer responsibility.

- Finally, monitor progress recording variances from design, documenting, reviewing and implementing change requests.



SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX O

STATIC WEB PAGES

Section O: Static Web Pages

New Development Deliverables

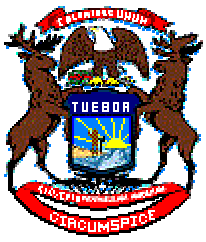
Static Web Development		
Definition: These deliverables are for Static web sites only. A static web site is defined as one that does not display any data drawn from a database, or interact with the user. For non-static sites, full project methodology must be used.		
Deliverable (section)	Definition	Tasks (WBS)
Ball Park Estimate	General estimate of the costs and times do complete the project. (Answers the question: Is this a peanut or an elephant?)	• Complete Ball Park Estimate
Web Requirements Document		
Project Description	General Description and background of project	• Include a general description of the request
Objectives of project	Include all user agency goals that must be met by the project deliverables	• Draft and then finalize project objectives to include training, administrative procedures. ITSD deliverables • Perform final proof and edit
Requirements	Assumptions as to how the project objectives will be accomplished within the scope.	• Review objectives, identify at least one requirement for each objective *Special consideration should be addressed to the following base project requirements(this is not an exhaustive list):* • Browser requirements • Look and Feel • Art Work • Web Statistics reporting • Interface requirements • Security • Search capabilities • # of prototypes

New Development Deliverables

Static Web Development		
Definition: These deliverables are for Static web sites only. A static web site is defined as one that does not display any data drawn from a database, or interact with the user. For non-static sites, full project methodology must be used.		
Deliverable (section)	Definition	Tasks (WBS)
Scope	Define the boundaries of the project	• Identify known constraints, mandates, frequencies, capabilities, etc.
Web page flow diagram	Web page flow for the major functional areas of the new site	• Develop Web Page flow diagram
Web page flow narrative	Narrative of the web page flow functional sections	• Develop narrative
Impact Analysis • Organizational • System	Organizational Impact: Describe the impact the web site will have on the organization. Include customer impact, cost/time/resource impact, software/hardware needs, training needs. System Impact: Describe the technical impact the new system/application will have. Include implementation requirements	Organizational • Write organizational impact analysis System • Research impact the new application will have on the system
E-MICHIGAN Approval	Approval process has not yet been formulated.	• Forward Design to E-Michigan for approval
High Level Estimate	Detailed cost analysis based on the defined requirements in the Web Requirements document	• Complete the High Level Estimate

New Development Deliverables

Static Web Development		
Definition: These deliverables are for Static web sites only. A static web site is defined as one that does not display any data drawn from a database, or interact with the user. For non-static sites, full project methodology must be used.		
Deliverable (section)	Definition	Tasks (WBS)
Development/Prototype Web Site	Creation of Web Pages	• Develop web pages for Web Site in development • Present prototype to customer in test
User approval of Web Site	Qualified Web Page development	• Move files to production Web Site
Proper storage of request, Web Site files, and approval in VSS	All documentation and Web Site moved to VSS	• Move required documentation to VSS



SYSTEMS DEVELOPMENT LIFECYCLE

APPENDIX T

BIBLIOGRAPHY

Appendix T: Bibliography

Section 2 Lifecycle Model

The following materials are references for the Lifecycle model section:

1. Booch, G., *Object-Oriented Analysis and Design,* 2nd edition, Benjamin Cummings, 1994.
2. Budd, T., *An Introduction to Object-Oriented Programming,* 2nd edition, Addison-Wesley, 1996.
3. Carnegie Mellon University, Software Engineering Institute, *Capability Maturity Model: Guidelines for Improving the Software Process*s, Addison Wesley Longman, Inc., 1994
4. Carney, D, & Oberndorf, P. "The Commandments of COTS: Still Searching for the Promised Land." Crosstalk 10, 5 (May 1997): 25-30.http://www.stsc.hill.af.mil/crosstalk/
5. Carney, D. Assembling Large Systems from COTS Components: Opportunities, Cautions, and Complexities [online].
6. Jacobson, I., *Object-Oriented Software Engineerin*g, Addison-Wesley, 1992.
7. Meyers, Craig & Oberndorf, Tricia. Open Systems: The Promises and the Pitfalls. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997.
8. Open Systems Joint Task Force Baseline Study, 1996 [online].
9. Pressman, Roger S., *Software Engineering - A Practitioner's Approach,* 4th edition, McGraw-Hill Companies, Inc., 1997.
10. Yourdon, Edward; Structured Walkthroughs, second edition, YOURDON Inc., New York, 1978.

Section 4 Planning Stage

The following materials are references for the Planning Phase section:

1. Carnegie Mellon University, Software Engineering Institute, *CapabilityMaturity Model: Guidelines for Improving the Software Process*s, AddisonWesley Longman, Inc., 1994
2. Project Management Institute, *A Guide to the Project Management Bodyof Knowledg*e, Pennsylvania, 1996. http://www.pmi.org/
3. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes,* IEEE Std 1074-1991, New York, 1992.
4. U.S. Department of Energy, *Chief Information Officer Partnering Team Project Management Process Guid*e, May 1998 (draft).
5. U.S. Department of Energy, *DOE/NV Software Management Plan,* Nevada Operations Office, May 1991.
6. U.S. Department of Energy, *Software Management Guide,* DOE/AD-0028,1992.

Section 5 Requirements Definition Phase

The following materials are references for the Requirements Definition Phase section:

1. Bramucci, Wilma, "Systems Development Software," *Faulkner Technical Reports, Inc*., June 1989. pp. 1-7.
2. Carnegie Mellon University, Software Engineering Institute, *Capability Maturity Model: Guidelines for Improving the Software Proces*s, Addison Wesley Longman, Inc., 1994
3. Fairley, Richard E., *Software Engineering Concepts,* McGraw-Hill Book Company, New York.
4. *Software Engineering Handboo*k, Chapter 4, Software Requirements Analysis.
5. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Requirements Specifications,* ANSI/IEEE Std 830-1984, New York, 1984.
6. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes,* IEEE Std 1074-1991, New York, 1992.
7. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Verification and Validation Plans,* ANSI/IEEE Std 1012-1986, New York, 1986.
8. U.S. Department of Commerce, National Bureau of Standards, *Guideline for Planning and Management of Database Applications,* Federal Information Processing Standards Publication 77, 1980. pp. 10-23.
9. U.S. Department of Energy, Software Quality Assurance Subcommittee, *Guidelines for Software Requirements Managemen*t, July 1998

Section 7 System Design Phase

The following materials are references for the System Design Phase section:

1. Barker, Richard, *CASE*METHOD,* Tasks and Deliverables, 1990. pp. 5-10 and 5-11.
2. Bramucci, Wilma, "Systems Development Software," *Faulkner Technical Reports, Inc*., June 1989. pp. 1-7.
3. Gane, Chris and Sarson, Trish, *Structured Systems Analysis: Tools and Techniqu*es, Prentice-Hall, Inc., Englewood Cliffs, 1979.
4. *Software Engineering Handbook* Chapter 5, Software Design; and Chapter 7, Software Testing.
5. *Software System Testing and Quality Assuranc*e, Chapter 5, Integration.
6. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Design Descriptions,* IEEE Std 1016.1-1993, New York, 1993.
7. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Requirements Specifications,* ANSI/IEEE Std 830-1984, New York, 1984.
8. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Recommended Practice for Software Design Descriptions,* IEEE Std 1016-1987, New York, 1987.
9. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes,* IEEE Std 1074-1991, New York, 1992.
10. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Verification and Validation Plans,* ANSI/IEEE Std 1012-1986, New York, 1986.
11. U.S. Department of Defense, Military Standard, *Defense System Software Development,* MIL-STD-2167A, 1988. pp. 27-29.
12. U.S. Department of Defense, Military Standard, *Specification Practices,* MIL-STD-490 B, 1986. pp. 47-50.
13. U.S. Department of Defense, Military Standard, *Technical Reviews and Audits for Systems, Equipments, and Computer Software,* MIL-STD-1521 B, 1985. pp.5, 33-52.
14. U.S. Department of Energy, *Automated Data Processing Systems Development Methodology, Volume 1,* K/CSD/INF/86-3, Vol.1, R3, prepared under contract by Martin Marietta Energy Systems, Inc. at Oak Ridge National Laboratory, August 1987.
15. U.S. Department of Energy, *Hanford Site Data Administration Guid*e, RLO-ADP-1, July 1989.
16. U.S. Department of Energy, *Software Management Guide,* DOE/AD-0028, 1992.

17. U.S. Department of Energy, Nevada Operations Office, *Software Management Plan,* May 1991.
18. U.S. Social Security Administration, Office of Systems, *Software Engineering Technology (SET) Manual,* Volume 1, 1990. Part 40 Design Stage.
19. Yourdon, inc., *Management Overview of Real-Time Structured Analysis and Desig*n, Edition 3.0, January 1984.

Section 8 Programming Phase

The following materials are references for the Programming Phase Section:

1. *Software Engineering Handboo*k, Chapter 7, Software Testing.
2. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes,* IEEE Std 1074-1991, New York, 1992.
3. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software User Documentation,* IEEE Std 1063-1987, New York, 1988.
4. U.S. Department of Energy, *Automated Data Processing Systems Development Methodology, Volume 1,* K/CSD/INF/86-3, Vol.1, R3, prepared under contract by Martin Marietta Energy Systems, Inc. at Oak Ridge National Laboratory, August 1987.
5. U.S. Department of Energy, *DOE/NV Software Management Plan,* Nevada Operations Office, May 1991.
6. U.S. Department of Justice, Immigration and Naturalization Service, *System Development Life Cycle Standards Manual,* 1991.

Section 10 Installation and Acceptance Phase

The following materials are references for the Installation and Acceptance Phase section:

1. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes,* IEEE Std 1074-1991, New York, 1992.
2. U.S. Department of Commerce, National Institute of Standards and Technology, *Guide to Software Acceptance,* 500-180, Washington, D.C., 1990.
3. U.S. Department of Energy, *Automated Data Processing Systems Development Methodology, Volume 1,* K/CSD/INF/86-3, Vol.1, R3, prepared under contract by Martin Marietta Energy Systems, Inc, at Oak Ridge National Laboratory, August 1987.
4. U.S. Department of Energy, *DOE/NV Software Management Plan,* Nevada Operations Office, May 1991.
5. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Concepts and Procedures Manual,* 1988.
6. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Standards Manual,* 1988.